# THE WINDOW CORNER ALGORITHM FOR POLYHEDRAL ROBOT AND ASSEMBLY SEQUENCE PATH PLANNING

NAGW-1333

by

S.S. Krishnan and A.C. Sanderson

Rensselaer Polytechnic Institute
NYS Center for Advanced Technology in Automation and Robotics
Electrical, Computer, and Systems Engineering Department
Troy, New York 12180-3590

May 1992

CIRSSE REPORT #117

Submitted to IEEE Transactions on
Robotics and Automation



# THE WINDOW CORNER ALGORITHM FOR POLYHEDRAL ROBOT AND ASSEMBLY SEQUENCE PATH PLANNING

S.S. Krishnan and A.C. Sanderson

New York State Center for Advanced Technology in Automation and Robotics
and Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180
May 3, 1992

**Abstract**

We present a complete and exact approach for planning a feasible path for an arbitrary (non-convex) polyhedral subassembly (robot), $R$, translating amongst arbitrary (non-convex) polyhedral obstacles, $\Phi$. This is the *FeasiblePath* problem. Our algorithm can be used to determine whether a pair of non-convex subassemblies can be separated by a sequence of fine-motion and, possibly, contact-constrained translations, even when the amount of free space tends to be severly limited. The *ShortestPath* problem entails finding the shortest feasible path. We present the *2D Window Corner* ($\mathcal{WC}$) algorithm, which is a novel solution to both problems for the case of translational paths in two dimensions (the robot or obstacles are not required to be simply connected), followed by the *3D Window Corner* algorithm, for feasible paths.

The *Polyhedral Cone Representation* ($\mathcal{PCR}$) is introduced to efficiently represent the geometric constraints between the boundaries of $R$ and $\Phi$, and as a basis for fast collision avoidance checks. We introduce the concept of *Window Corners* in the *Polyhedral Cone Representation* ($\mathcal{PCR}$) and this results in reducing the search space. The *Polyhedral Cone Obstacle Representation* ($\mathcal{PCOR}$) transforms the problem (in $E^2$ and $E^3$), into that of a point moving amongst a collection, $O(m)$, of convex obstacles. The $\mathcal{PCOR}$ is a constructive representation of the set of contact configurations between $R$ and $\Phi$. In comparison, the worst-case number of vertices in the B-rep. of the *C-space* obstacles is $O(m^2)$ in 2D and $O(m^3)$ in 3D, and requires the same order of time to construct. In addition, contact-constrained, feasible motions, which lie on surfaces without interior points will not be accessible in a typical B-rep. of the *C-space* obstacles.

In two dimensions, a *Window Graph* with $k$ nodes and $O(k^2)$ edges, is built, and the shortest path found, in $O(km\log(m))$ time where $m$ is the product of the number of vertices describing the robot and obstacles respectively, $k = O(m)$ in the worst case and $1 \leq k \leq (m+2)$. The nodes of the $\mathcal{W}$-Graph correspond to a subset of the convex vertices of the *C-space* representation. In three dimensions, the *Window Corner* ($\mathcal{WC}$) algorithm constructs a *Window Edge* ($\mathcal{WE}$)-*Tree* and solves the *FeasiblePath* problem, in worst-case time $O(m^5)$. Our approach finds *Window Edges*, which correspond to a subset of the convex edges of the *C-space* boundary, and traces a path through *Window Corners*.

We have implemented and tested the $\mathcal{WC}$ algorithms and examples from robot-independent assembly sequence planning are presented in this paper. The 3D $\mathcal{WC}$ algorithm searches a finite search space, made possible by the use of *Window Corners*, and both the 2D and 3D versions are attractive for rapid implementation in robotic and assembly sequence path planning domains.

ORIGINAL PAGE IS
OF POOR QUALITY



# 1 Introduction

We present an algorithm for solving the *FeasiblePath* problem for the case of a polyhedral subassembly (robot), $R$, translating amongst polyhedral obstacles, $\Phi$, in order to attain a pre-specified goal configuration. The robot and obstacles are described by their bounding faces. The algorithm is complete and exact, in that it can find fine-motion paths and feasible translations, which may be entirely constrained by contacts. Our algorithm can be used, during robot-independent Assembly Sequence Planning [4], to determine the separability (or assemblability) of a pair of non-convex polyhedral subassemblies. In this case, the "robot" and "obstacles" each correspond to a subassembly. The robot corresponds to the moving subassembly. Krishnan and Sanderson [5] study the assembly path planning problem and present algorithms to analytically determine all feasible straight-line directions for automatic assembly, by reasoning about the geometric constraints between subassemblies. Wilson and Rit [18] describe rules to reason about the feasibility of a disassembly, based on the outcome of previous disassembly tests.

In an Assembly Sequence Planning (ASP) environment, the amount of free space tends to be severely limited. While many path planners oriented towards mobile robot navigation have simplified the problem by modelling the robot as a disc, the method described here considers an arbitrary polyhedral, constrained workspace which will render the disc-model inadequate. Algorithms which achieve fast performance by using approximate representations of the obstacles or the free space are no longer guaranteed to find a feasible path if one exists. In our approach, the robot and obstacles are described as a collection of polygons (represented by edges) or polyhedrons (represented by faces), in two and three dimensions, respectively. Note that the description is not required to be the precise boundary representation of the objects. We require that the algorithm be correct and complete, in that it should be guaranteed to find a path if one exists, and that every path reported should indeed be feasible.

We solve the *FeasiblePath* problem in a hierachical manner. We first determine whether some incremental motion is possible, by analyzing all the contacts between the robot and the obstacles. We then test for the existence of a straight-line path to separate a pair of subassemblies. Finally, the algorithm searches for paths with multiple translations. Determining the feasibility of incremental motion, requires analyzing all the contacts between $R$ and $\Phi$, and can be done rapidly. This test is indispensable, since it serves as a necessary condition for the existence of a feasible path. We expect that searching for a straight-line translational path, can be accomplished by a fast algorithm, while searching for paths with multiple translations and for paths with translations and/or rotations, would require increasing amounts of time. In fact, the computation times progressively increase by several orders of magnitude, mainly due to the increased dimensionality of the problem. The complexity of the feasible assembly path serves as a component of an *evaluation function*, which is used to guide the assembly sequence planning process, in order to identify the most optimal (according to specified criteria) assembly sequence plan. Also, designing an assembly which only requires simple (e.g. straight-line) translational motions for the parts, would significantly reduce costs and increase throughput of an automated assembly system. Such issues are of vital importance in the emerging area of concurrent engineering, where "Design For Assembly" (DFA) and "Design For Manufacture" (DFM) are design approaches which

seek to improve design quality and assembly throughput, while reducing manufacturing costs.

Previous works on planar path planning have, in general, addressed variants of the *ShortestPath* problem, which requires finding an optimal path, usually with an Euclidean distance metric. The *Configuration (C)-space* formulation of the problem has been extensively used in the literature (Lozano-Pérez et al [9], Lozano-Pérez [10]). In C-space, the moving object is shrunk to a point while the obstacles are simultaneously enlarged, and this conceptually simplifies the problem. Constructing the explicit B-rep. of the C-space obstacles has both space and time complexity which is exponential in the d-o-f of $R$ and polynomial in the complexity of the workspace.

In two dimensions, a *Window Graph* with $k$ nodes and $O(k^2)$ edges, is built in $O(km\log(m))$ time where $m$ is the product of the number of vertices describing the robot and obstacles respectively. $k = O(m)$ in the worst case and $1 \leq k \leq (m+2)$. The $\mathcal{WC} - \mathcal{PCOR}$ algorithm utilizes the concept of a *Polyhedral Cone Obstacle Representation* ($\mathcal{PCOR}$) which transforms the problem, in $O(m)$ time, into that of a point moving amongst a collection, $O(m)$, of convex parallelogram shaped obstacles. In comparison, the space complexity of *Configuration (C)-space* is $O(m^2)$ (Latombe[7], Sharir[13]). The $\mathcal{PCOR}$ consists of an $O(m)$ edges, each of which may be broken up, in the worst-case, into an $O(m)$ segments, by the $O(m)$ convex obstacles. Thus, in the worst-case, the B-Rep. of the C-space obstacles has an $O(m^2)$ vertices. But, we have shown that the vertices of the $O(m)$ convex obstacles form the reduced search space for solving the *FeasiblePath* problem. Any new vertices in the B-Rep. of the C-space obstacles will be non-convex vertices, and are not required to be visited by the *FeasiblePath* search algorithm.

Let $n$ be the number of vertices in the *C-space* representation of the problem. $O(n^2)$ time algorithms are known (Welzl[14], Asano et al [1]) to solve the *C-space* representation of the problem, these algorithms have certain assumptions such as non-intersecting edges (or polygons) in the input. An output sensitive algorithm was presented by Ghosh et al [2] while Liu et al [8] plan the motion of a circular disc. The previously fastest algorithm was an $O(n^2\log(n))$ algorithm using a sweep-line technique (Sharir et al [12]). The $\mathcal{WC} - \mathcal{PCOR}$ algorithm is expected to improve on any *C-space* based method since, in the worst case, $n = O(m^2)$, and these methods would spend an $O(m^2)$ time in first constructing a boundary rep. of *C-space*. Further, any *C-space* based algorithm is applicable to the $\mathcal{PCOR}$, since the *C-space* boundary can be computed from the $\mathcal{PCOR}$ in $O(m^2)$ time. From a *C-space* perspective, the $\mathcal{W}$-Graph nodes correspond to *a subset of the convex vertices* in the *C-space* representation.

In three dimensions, the *ShortestPath* problem requires exponential time (Sharir et al[12]). Papadimitriou[11] presented an approximation algorithm for finding the shortest path. Hwang and Ahuja [17] present an algorithm, based on a potential field rep. of the obstacles, for finding paths which may include rotations. They solve the problem in 3D world space by restricting a search of the orientation space to cluttered regions. We introduce the *Polyhedral Cone Obstacle Representation* ($\mathcal{PCOR}$) which transforms the problem into that of a point moving amongst a collection, $O(m)$, of convex obstacles. The $\mathcal{PCOR}$ is a constructive represenatation of the set of contact configurations between the robot and the obstacles. In comparison, the worst-case space complexity of the boundary rep. of *C-space* is $O(m^3)$ in three dimensions (Latombe [5], Sharir [10]), and requires the same order of time

to construct. There are an $O(m)$ faces in the $\mathcal{PCOR}$. Each face is broken into sub-faces, due to intersections with the $O(m)$ convex obstacles in the $\mathcal{PCOR}$. In the worst case, each face may be divided into an $O(m^2)$ unconnected sub-faces. The boundary of *C-space* is the union of a subset of these sub-faces, resulting in an $O(m^3)$ worst-case space complexity for the boundary representation. In regions where motion is completely constrained by contacts, the feasible translations may lie on surfaces with no interior points. Such surfaces can be inaccessible in a typical *C-space* boundary representation. Our *3D Window Corner* algorithm solves the *FeasiblePath* problem in polynomial ($O(m^5)$) time by tracing a path through *Window Corners* located on *Window Edges*. This renders solving *FeasiblePath* problems, preferable over solving for the shortest path for many robotic applications. From a *C-space* perspective, the *Window Edges* correspond to *a subset of the convex edges* on the boundary of a *C-space* representation.

Further, the *FeasiblePath* problem is preferable in many robotic domains. The *FeasiblePath* problem differs from the *ShortestPath* problem, since it entails planning a *feasible* path which may not necessarily be an *optimal* path. Although, the *FeasiblePath* problem can be reduced to the *ShortestPath* problem, it's distinguishing features provide the motivation to develop efficient solutions.

The *FeasiblePath* problem appears in many robotics-related domains. In many robotic settings, it would be preferable to find some feasible path quickly and commence operations, rather than spend large amounts of time in planning an optimal path. Planning a feasible path is vital to generating motions for automatic assembly, telerobotic maintenance and repair in manufacturing or in hazardous situations such as in nuclear-radiation zones, underwater or in deep-space exploration. Building sensor-integrated autonomous robot systems is a focus of current research in robotics. Such intelligent robots have to frequently plan their movements and reach certain goal configurations, in order to accomplish various tasks. Typically, the robot would have to self-navigate among a set of obstacles in it's environment, hence requiring solutions to *FeasiblePath* problems. An emerging area is that of concurrent engineering. Here, automated product design and assembly sequencing give rise to a very large number of tests to determine the assemblability (or separability) of two subassemblies subject to geometric constraints (Baldwin et al [19], Homem de Mello and Sanderson [3,4]). Each of these combinatorially large number of tests, is a *FeasiblePath* problem,

Krishnan and Sanderson [6], have introduced the *Polyhedral Cone Representation*, *Polyhedral Cone Obstacle Representation* and *Window Corners* for the two-dimensional version of the *FeasiblePath* problem. They proposed the *2D Window Corner* algorithm, which solved the planar *FeasiblePath* and *ShortestPath* problems by tracing a path through *Window Corners*. The above features resulted in a lower time complexity than previous works which first construct the *C-space*. In this paper, *Window Edges* are introduced and the $\mathcal{PCR}$ and $\mathcal{PCOR}$ are defined for a three-dimensional environment. The resulting *3D Window Corner* algorithm gives a greater relative improvement in time complexity over algorithms which construct the boundary of *C-space* in the three-dimensional case as compared to the planar version. The notations used in this paper have been kept consistent with those in [6]. Geometric translational constraints are converted into a Polyhedral Cone Representation of the problem. The concept of **Window Corners**, along with a **Window Corner Theorem**, is used to provide a finite and reduced solution space. Finally, the search ripples through a single connected component of reachable space containing the initial placement of the

robot.

In Section 2, we present the *Window Corner* 2D ($\mathcal{WC}$) algorithm in two versions. Subsection 1 introduces the definitions and terminology which will be used in this paper for the 2D algorithm. Subsection 2 defines the **Window Corners** and the important **Window Corner Theorem**. Subsection 3 describes the Polyhedral Cone Representation. In Subsection 4, the definition of a Polyhedral Cone Obstacle Representation ($\mathcal{PCOR}$), transforms the problem into that of a point moving amongst parallelogram shaped obstacles and forms the basis for the $\mathcal{PCOR}$ version. We also compare the $\mathcal{PCOR}$ to the *C-space* representation. In Subsection 5, the $\mathcal{PCR}$ and $\mathcal{PCOR}$ versions of the $\mathcal{WC}$ algorithm are presented, along with an analysis of their time complexity. An assembly path planning problem and it's solution are presented in Subsection 6.

Section 3 describes the $\mathcal{WC}$ algorithm in three dimensions, which follows the same logical development as the planar version. In Subsections 1 and 2, we introduce definitions, the concept of **Window Edges**, and the **Window Edge** and **Feasible Path** Theorems, followed by the definition of **Window Corners**. In Subsection 3, the Polyhedral Cone Representation of geometric constraints is introduced. Subsection 4 provides the Polyhedral Cone Obstacle Representation, which conceptually reduces the problem to that of a point moving amongst a collection of convex, parallelopiped-shaped obstacles. In Subsection 5, we describe a test to determine the feasibility of incremental motion based on all the contact constraints. Next, we test for the existence of any straight-line assembly path. The *3D Window Corner* algorithm for multiple-step paths, is then described, along with a discussion of it's time complexity. In Subsection 6, an assembly path planning example is provided to illustrate the working of the 3D $\mathcal{WC}$ algorithm. In Subsection 7, we show how the $\mathcal{PCOR}$ can be utilized to compute the minimum distance between the robot and obstacle boundaries. This algorithm conducts a search over only a *subset of the convex edges*, in terms of a *C-space* representation.

# 2 FeasiblePath and ShortestPath Problems in Two Dimensions

## 2.1 Preliminaries

Definitions:

- $\pi_V \triangleq$ a set of vertices, $\{v_i^\pi\}\ \forall\ i = 1, \ldots, v(\pi)$,
- $\pi_E \triangleq$ a set of edges, $\{e_i^\pi\}\ \forall\ i = 1, \ldots, e(\pi)$, ( without loss of clarity, we will sometimes refer to $\pi_E$ as the set of all points contained in these edges ), $\pi_V \subset \pi_E$,
- $\pi_R \triangleq$ a set of specified polygonal regions, that the edges may enclose, (and referring to a set of points contained in those regions),
- $\pi \triangleq \{x : x \in E^2 \ \wedge\ (x \in \pi_E \ \vee\ x \in \pi_R)\}$, (the set of all points contained in the polygonal regions, edges and vertices),
- $F \triangleq E^2 \setminus \pi$.

In the remainder of this paper, a point $q$ is said to be *straight-line reachable* from another point $p$, if the straight-line translation from $p$ to $q$ is collision-free.

Given two points $p, q \in E^2$, $\vec{q}$ denotes the position vector corresponding to the point $q$, $\vec{pq} = \vec{q} - \vec{p}$, $\overline{pq}$ denotes the (interchangeably, the set of points in) line segment between $p$ and $q$, and $\|\overline{pq}\|$ denotes the length of the line segment from $p$ to $q$. We now introduce the following definitions. Let $p \in F$.

**Definition 1** *An $\alpha$-neighborhood of $p$ is defined as a set $\mathcal{N}_\alpha(p) = \{u : u \in E^2, \|\overline{pu}\| \leq \alpha\}$ where $\alpha$ is a small, positive, real number.*

**Definition 2** *A $\beta$-neighborhood of $p$ is defined as a set $\mathcal{N}_\beta(p) = \{u : u \in \pi, \|\overline{pu}\| \leq \beta\}$ where $\beta$ is a small, positive, real number.*

**Definition 3** *A point $q \in \pi$ is a* **Window Corner** *w.r.t. $p$, written as $\mathcal{WC}(q, p)$, iff:*

1. *$q \in \pi_V$ and*
2. *there exists a small $\beta$ and a normal, $\vec{n}$, to $\overline{pq}$, such that $\vec{n} \cdot \vec{pv} \geq 0 \quad \forall\, v \in \mathcal{N}_\beta(q)$.*

**Definition 4** *A half-line emanating from $p$ is called a ray, $r(p)$, w.r.t. $p$. The set of all rays from $p$ is denoted by $\rho(p)$.*

**Definition 5** *A ray, $r(p)$, is called a colliding ray, iff:*

1. *$r(p) \in \rho(p)$, and*
2. *$r(p)$ intersects exactly $l$ bounding edges of $\pi$ where $l \geq 1$.*

*The set of all colliding rays, w.r.t. $p$, is denoted by $\rho'(p)$.*

Let the discrete set of intersection points between $r(p)$ and $\pi_E$, in sequence from $p$, be denoted by $\overline{\mathcal{IP}(r(p), i)} \quad \forall\, i = 1, \ldots, l$ where $r(p) \in \rho'(p)$.

**Definition 6** *$\theta_i(p)$ is the $i$-th* **window ray** *w.r.t. $p$, iff:*

1. *$\theta_i(p) \in \rho'(p)$, and*
2. *$\mathcal{WC}(\mathcal{IP}(\theta_i(p), 1), p)$, i.e. the first point of intersection on the colliding ray is a* Window Corner *w.r.t. $p$.*

*Let $\Theta = \{\theta_i(p)\} \quad \forall\, i = 1, \ldots, w_r$.*

## 2.2 Windows and the Window Corner Theorem

For a given **window ray** $\theta_i(p)$: let $\mathcal{WC}(\mathcal{IP}(\theta_i(p), j), p) \quad \forall\, j = 1, \ldots, r$ where $1 \leq r \leq l$. We now define **Windows** which are a (possibly unbounded) set of contiguous, collinear points on a **window ray**, as follows:

**Definition 7** *Let $q_j = \mathcal{IP}(\theta_i(p), j) \quad \forall\, j = 1, \ldots, l$, and let the $k$-th* **Window** *on $\theta_i(p)$ be denoted by $\mathcal{W}(\theta_i(p), k)$.*

1. *If $r = l$:*
   (a) *If $r = 1$, then $\mathcal{W}(\theta_i(p), r) = \{b : \vec{b} = (\lambda)p\vec{q}_r, 1 \leq \lambda\}$.*
   (b) *else, $\mathcal{W}(\theta_i(p), k) = \{b : \vec{b} = (1-\lambda)\vec{q_k} + \lambda\vec{q_{k+1}}, 0 \leq \lambda \leq 1, 1 \leq k \leq (r-1)\}$ and $\mathcal{W}(\theta_i(p), r) = \{b : \vec{b} = (\lambda)p\vec{q}_r, 1 \leq \lambda\}$.*

2. *If $r < l$: The first $(r-1)$* **windows** *are defined as in (1(b)) above. $\mathcal{W}(\theta_i(p), r) = \{b : \vec{b} = (1-\lambda)\vec{q_r} + \lambda\vec{q_{r+1}}, 0 \leq \lambda \leq 1\}$.*

**Definition 8** *A collision-free path, $\Psi$, from some initial position $S$ to some goal position $G$, is defined as a sequence of $n$ points $\psi_1\psi_2, \ldots, \psi_n$ such that $\overline{\psi_i\psi_{i+1}} \cap \pi = \emptyset, \forall\, i = 1, \ldots, (n-1)$, and where $\psi_1$ corresponds to $S$ and $\psi_n$ corresponds to $G$.*

Assume that we are given two points $p, q \in F$, but $\overline{pq}$ is not collision-free i.e. $\overline{pq} \cap \pi \neq \emptyset$. Also assume that the shortest collision-free path from $p$ to $q$ is $p\omega q$, where $\omega \in E^2$, $\overline{p\omega} \cap \pi = \emptyset$, $\overline{\omega q} \cap \pi = \emptyset$. We now state the following theorem:

**Theorem 1 (Window Corner Theorem)** *If the shortest collision-free path from $p$ to $q$ changes direction at a single point $\omega$, then $\omega$ is a Window Corner w.r.t. $p$.*

**Proof:** The proof is presented in two parts.

1. $\omega \in \pi_V$: By contradiction. Let $((a \in (\mathcal{N}_\alpha(\omega)) \cap \overline{p\omega}) : (\|\overline{a\omega}\| = \alpha))$ and $((b \in (\mathcal{N}_\alpha(\omega)) \cap \overline{\omega q}) : (\|\overline{\omega b}\| = \alpha))$. If $(\omega \in F)$, see Figure 1(a), then $\exists\alpha : \mathcal{N}_\alpha(\omega) \cap \pi = \emptyset$, hence $\overline{ab} \cap \pi = \emptyset$. If $((\omega \in \pi_E) \wedge (\omega \notin \pi_V))$, see Figure 1(b) then $\exists\alpha : \overline{ab} \cap \pi = \emptyset$. But, $\|\overline{ab}\| < \|\overline{a\omega}\| + \|\overline{\omega b}\|$. Hence, $\|\overline{pa}\| + \|\overline{ab}\| + \|\overline{bq}\| < \|\overline{p\omega}\| + \|\overline{\omega q}\|$. Thus, if $\omega \notin \pi_V$, a shorter path can be found and so $\omega$ has to be an obstacle vertex.

2. Let $\omega$ be an obstacle vertex. We now show, by contradiction, that there should exist a normal $\vec{n}$, to $\overline{p\omega}$, for some small, positive $\beta$, such that
$$\vec{n} \cdot \vec{pv} \geq 0 \quad \forall \quad v \in \mathcal{N}_\beta(\omega). \tag{1}$$
A situation to the contrary is shown in Figure 1(c). In such a case, there exist some two points $a$ and $b$, as defined above, for some small, positive, real $\alpha$, such that $a, b \in N_\alpha(\omega)$ and $\|\overline{pa}\| + \|\overline{ab}\| + \|\overline{bq}\| < \|\overline{p\omega}\| + \|\overline{\omega q}\|$. Hence, a shorter path can be found and so $\omega$ should be such as to satisfy the condition (1) above.

The above (1) and (2) are necessary and sufficient conditions for $\omega$ to be a *Window Corner* w.r.t. $p$, i.e. $\mathcal{WC}(\omega, p)$, from Definition 3.

( In fact, the following condition should also be satisfied; $\vec{n} \cdot \vec{\omega q} > 0$.)

**Q.E.D.** □

**Corollary 1** *Given the shortest collision-free path, $\Psi$, as in Definition 8, every intermediate point on this path is a Window Corner with respect to it's predecessor on the path, i.e. $\mathcal{WC}(\psi_{i+1}, \psi_i)\ \forall\ i = 1, \ldots, (n-2)$ and $\mathcal{WC}(\psi_i, \psi_{i+1})\ \forall\ i = 2, \ldots, (n-1)$.*

**Proof:** For $\Psi$ to be the shortest collision-free path from $\psi_1$ to $\psi_n$, it is necessary that it represents the shortest collision-free path between any three successive points in $\Psi$. In other words, $\psi_i\psi_{i+1}\psi_{i+2}$ is the shortest collision-free path between $\psi_i$ and $\psi_{i+2}, \forall\, i = 1, \ldots, (n-2)$. From the **Window Corner Theorem**, it follows that $\mathcal{WC}(\psi_{i+1}, \psi_i)\ \forall\ i = 1, \ldots, (n-2)$ and $\mathcal{WC}(\psi_i, \psi_{i+1})\ \forall\ i = 2, \ldots, (n-1)$.

**Q.E.D.** □

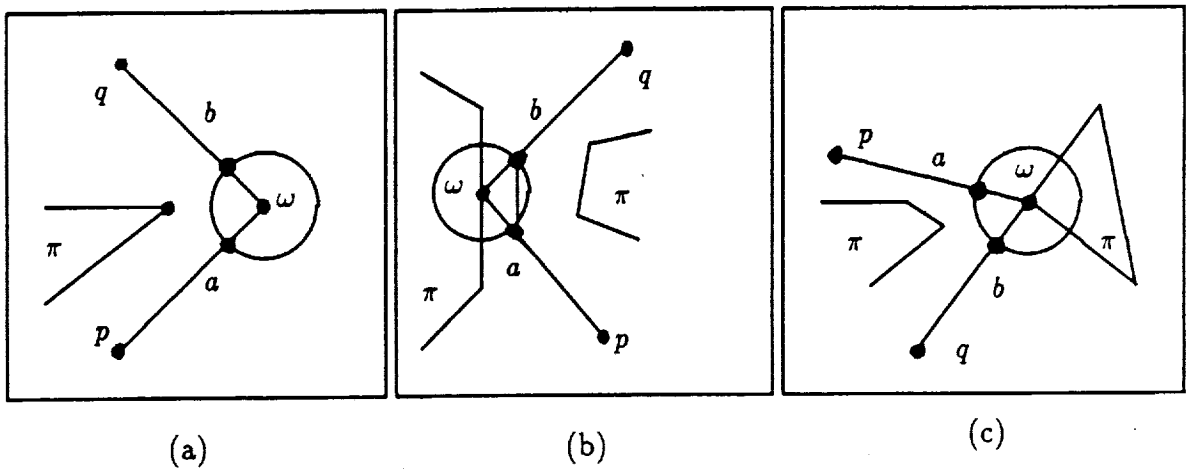


Figure 1: Proof of the Window Corner Theorem

## 2.3 Polyhedral Cone Representation ($\mathcal{PCR}$)

For the rest of the paper, we consider a robot, $R$, and obstacles, $\Phi$, to be described by:

- $RV \triangleq$ the set of robot vertices, $\{v_i^R\}\ \forall\ i = 1, \ldots, v(R)$,
- $RE \triangleq$ the set of robot edges, $\{e_i^R\}\ \forall\ i = 1, \ldots, e(R)$, (we will also refer to $RE$ as the set of points contained in these edges), $RV \subset RE$,
- $RR \triangleq$ a set of specified polygonal regions (or the points contained in them) that the robot edges may enclose,
- $R \triangleq \{x : x \in E^2 \wedge (x \in RE \vee x \in RR)\}$,
- $\Phi V \triangleq$ the set of obstacle vertices, $\{v_i^\Phi\}\ \forall\ i = 1, \ldots, v(\Phi)$,
- $\Phi E \triangleq$ the set of obstacle edges, $\{e_i^\Phi\}\ \forall\ i = 1, \ldots, e(\Phi)$, ( without loss of clarity, we will sometimes refer to $\Phi E$ as the set of all points contained in these edges ), $\Phi V \subset \Phi E$,
- $\Phi R \triangleq$ a set of specified polygonal regions (or the points contained in them) that the obstacle edges may enclose,
- $\Phi \triangleq \{x : x \in E^2 \wedge (x \in \Phi E \vee x \in \Phi R)\}$,
- $F \triangleq E^2 \setminus \Phi$.

Note that the robot and obstacles are basically a collection of (possibly unconnected) edges in a planar environment. In addition, $S$ and $G$ refer to the start and goal positions of

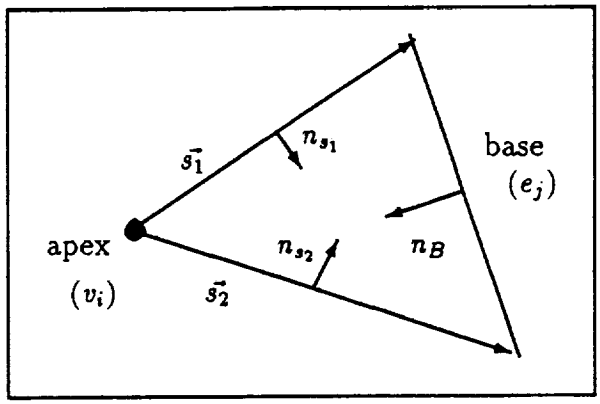


Figure 2: A *Vertex-Edge* constraint cone

some fixed reference point on the robot, respectively. Also, we define the space complexity of the problem by the following terms:

$$m = v(R)v(\Phi), \quad e(R)e(\Phi) = O(m), \quad v(R)e(\Phi) = O(m), \quad e(R)v(\Phi) = O(m). \tag{2}$$

The constraint on the translation of a vertex with respect to an edge can be represented as a *Vertex-Edge (VE) constraint cone*, defined as follows.

**Definition 9** *A Vertex-Edge constraint cone, $C_{VE}(v_i, e_j)$, between a vertex, $v_i$, and an edge, $e_j$, consists of two bounding vectors, $\vec{s_1}$ and $\vec{s_2}$, drawn from the vertex to the end points of the edge, and a base $e_j$, which is the line segment joining the ends of the two bounding vectors. A VE cone, consisting of an apex, a base and two bounding vectors, is shown in Figure 2.*

A *VE cone* can be unambiguously represented by three unit vectors $\vec{n_{s_1}}, \vec{n_{s_2}}, \vec{n_B}$, (which are the normals to the two bounding vectors and the base of the *VE cone* respectively) and one of the bounding vectors, say, $\vec{s_1}$. These four vectors are collectively called the *normal representation* of the *VE cone.*

**Definition 10** *The normal representation of the negation of a VE cone, denoted by $-C_{VE}(v_i, e_j)$, is obtained by reversing the direction of each of the vectors in the normal representation of the VE cone.*

We are now ready to define the Polyhedral Cone Representation, $\mathcal{PCR}$, as follows. In the two-dimensional context, the constraint cones are actually polygonal, but will be extended to polyhedral cones in three-dimensions. Given a robot, $R$, at some position and orientation in $E^2$, and a set of obstacles, $\Phi$:

**Definition 11** *The set of all VE cones between the robot and obstacles, is called the Polyhedral Cone Representation, $\mathcal{PCR}(R, \Phi)$, or simply $\mathcal{PCR}$, i.e.*

$$\mathcal{PCR} = \{C_{VE}(v_i^R, e_j^\Phi) \forall\ i, j\} \cup \{-C_{VE}(v_k^\Phi, e_l^R) \forall\ k, l\}. \tag{3}$$

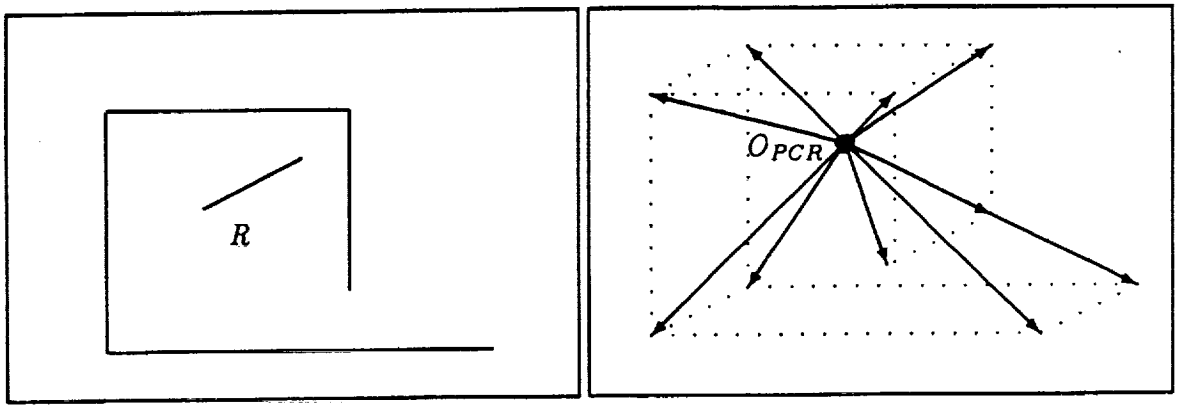


Figure 3: (a) Robot, R, and obstacles. (b) The PCR

The origin of the $\mathcal{PCR}$ is denoted by $O_{\mathcal{PCR}}$. An example of a $\mathcal{PCR}$ is shown graphically in Figure 3.

**Definition 12** *A vector $\vec{v} \in R^2$ is said to be infeasible w.r.t. some* VE cone, $C_j$, *written $INFEAS(\vec{v}, C_j)$, iff:*

1. $\vec{v} \cdot \vec{n_{s_1}} \geq 0$,
2. $\vec{v} \cdot \vec{n_{s_2}} \geq 0$ *and*
3. $\vec{n_B} \cdot (\vec{v} - \vec{s_1}) < 0$.

*If semi-free paths are allowed, conditions 1 and 2 are strict inequalities.*

**Definition 13** *Vector $\vec{v}$ is infeasible w.r.t. the $\mathcal{PCR}$, i.e.*

$$INFEAS(\vec{v}, \mathcal{PCR}) \Leftarrow if\ \exists C_j : (C_j \in \mathcal{PCR}) \wedge (INFEAS(\vec{v}, C_j)). \tag{4}$$

*A vector that is not infeasible w.r.t the $\mathcal{PCR}$, is said to be feasible w.r.t. the $\mathcal{PCR}$, $FEAS(\vec{v}, \mathcal{PCR})$.*

**Theorem 2** *A translation of the robot by some vector $\vec{v}$, from it's current position is guaranteed to be collision-free iff $FEAS(\vec{v}, \mathcal{PCR})$.*

**Proof:** We know from Definitions 12 and 13 that if $FEAS(\vec{v}, \mathcal{PCR})$, then $\vec{v}$ does not pierce the base of any constraint cone. From Definition 9, this implies that no robot (obstacle) vertex collides with an obstacle (robot) edge. These are necessary and sufficient conditions for any translation among planar polygonal objects to be collision-free, and hence the proof.

**Q.E.D.** □

A vector is infeasible w.r.t. the $\mathcal{PCR}$ iff it pierces the base of any *VE cone*. Hence, for the motion of $O_{\mathcal{PCR}}$, we can consider the bases of the *VE cones* to be obstacle edges.

We know from Definition 3 that every *Window Corner* is the end point of some obstacle edge. Therefore, the *Window Corners* with respect to $O_{\mathcal{PCR}}$, are the vertices of the *VE cone* bases. An inspection of the *VE cone* from Definition 9 shows that these base vertices correspond to a contact between some robot vertex and an obstacle vertex. Since, any of these base-vertices, and only these base-vertices, can be *Window Corners* w.r.t. $O_{\mathcal{PCR}}$, these vertices form the set of *candidate Window Corners*, i.e.

$$\mathcal{CWC} = \{(v_i^R, v_j^\Phi) \; \forall \; i, j\} \quad or \tag{5}$$

$$\mathcal{WC}(c, O_{\mathcal{PCR}}) \Rightarrow (\exists (v_i^R, v_j^\Phi) \in \mathcal{CWC} : \vec{c} = (\vec{v_j^\Phi} - \vec{v_i^R})) \tag{6}$$

Each element of $\mathcal{CWC}$ corresponds to a labelling of a base-vertex of some *VE cone*. But, in the $\mathcal{CWC}$, the vertices are labelled as representing the contact between some robot vertex and some obstacle vertex. Thus, the search space to find the *Window Corners* w.r.t. $O_{\mathcal{PCR}}$, is of size $m$, even though there are physically, at least $2m$ base-vertices in the $\mathcal{PCR}$.

## 2.4 Polyhedral Cone Obstacle Representation ($\mathcal{PCOR}$)

Given one edge each on the robot, $i \in RE$, and obstacle, $j \in \Phi E$, with vertices $v_1, v_2$ and $v_3, v_4$ respectively. It can be observed that the base-edges of each of the four *VE cones* between $i$ and $j$, enclose a parallelogram-shaped region, which is entirely unreachable by a collision-free translation from $O_{\mathcal{PCR}}$.

**Definition 14** *A Polyhedral Cone Obstacle between two edges, $i$ and $j$, denoted by $\mathcal{PCO}(i,j)$, is the parallelogram defined by the four vertices whose position vectors are as follows: $(\vec{v_3} - \vec{v_1}), (\vec{v_3} - \vec{v_2}), (\vec{v_4} - \vec{v_1}), (\vec{v_4} - \vec{v_2})$. We will sometimes refer to a $\mathcal{PCO}$ as the set of all points contained in the parallelogram.*

**Definition 15** *The set of all $\mathcal{PCO}$s computed between robot and obstacle edges, at $S$, is called the Polyhedral Cone Obstacle Representation ($\mathcal{PCOR}$) of the problem;*

$$\mathcal{PCOR} = \{\mathcal{PCO}(i,j)\} \forall \; i = 1, \ldots, e(R) \quad \forall \; j = 1, \ldots, e(\Phi) \tag{7}$$

*We also refer to $\mathcal{PCOR}$ as the set of all points contained in the constituent $\mathcal{PCO}$s.*

It follows from the above that:

$$if \;\; \exists \mathcal{PCO}(i,j) : v \in \mathcal{PCO}(i,j) \Rightarrow INFEAS(\vec{v}, \mathcal{PCR}) \tag{8}$$

Since any point that is unreachable from $O_{\mathcal{PCR}}$ lies in the shadow of some $\mathcal{PCO}$, all infeasible positions (corresponding to boundary intersections) of the robot are contained in the $\mathcal{PCOR}$. An example of the $\mathcal{PCOR}$ is drawn in Figure 4. Let the origin of the $\mathcal{PCOR}$ be $O_{\mathcal{PCOR}}$; it corresponds to the start position of the robot. In the $\mathcal{PCOR}$, the various $\mathcal{PCO}$s form the obstacles for the motion of a point, henceforth called the point-robot $r$, from $O_{\mathcal{PCR}}$ to a point whose position vector is $\vec{G} - \vec{S}$.

**Theorem 3** *Finding a shortest collision-free multistep translation path, $\Psi$, for the point-robot from position $O_{\mathcal{PCR}}$ to position $\vec{G} - \vec{S}$, in the $\mathcal{PCOR}$, such that $\overline{\psi_i \psi_{i+1}} \cap \mathcal{PCOR} = \emptyset, \forall \; i = 1, \ldots, (n-1)$, is also a solution to the original problem.*

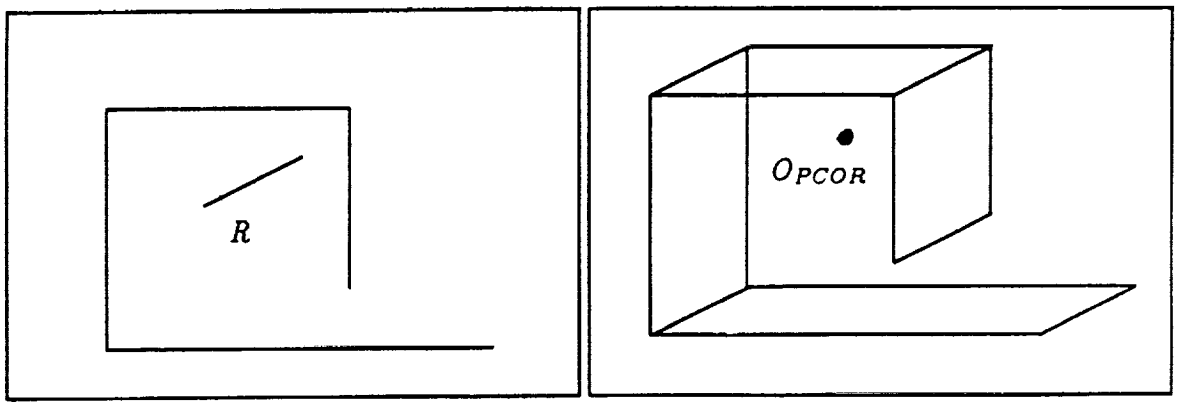


Figure 4: (a) Robot, R, and obstacles. (b) The PCOR

**Proof:** It is easily verified that the boundary edges of $\mathcal{PCO}(i,j)$ are the bases of the four *VE cones* between robot edge $i$ and obstacle edge $j$. In addition, each $\mathcal{PCO}$ is a parallelogram-shaped obstacle and is convex. Avoiding each $\mathcal{PCO}$ is equivalent to avoiding a collison between all pairs of robot and obstacle edges, since all vertex-edge collisions are avoided, from (8). This is because all infeasible points are contained within the $\mathcal{PCOR}$. Alternatively, the $\mathcal{PCO}s$ correspond to positions of the point-robot at which $RE \cap \Phi E \neq \emptyset$. The goal position for the point-robot is simply found to be $\vec{G} - \vec{S}$. $\Psi$, therefore provides the relative coordinates for the locus of any point on the robot $R$, and hence the solution to the original problem.

**Q.E.D.** □

### 2.4.1 A Comparison of $\mathcal{PCOR}$ and *C-space*

Hence, in the $\mathcal{PCOR}$, the problem is to find a collision-free path for a point-robot moving from the start position,$O_{\mathcal{PCOR}}$, to a goal position, $\vec{G} - \vec{S}$, among a set, $O(m)$, of parallelogram shaped, convex obstacles ($\mathcal{PCO}s$). It is interesting, at this juncture, to compare the $\mathcal{PCOR}$ with the *C-space* representation. The $\mathcal{PCOR}$ is a convex (parallelopiped shaped) constructive representation (superset) of the *C-space* obstacle boundaries. Collectively, the $\mathcal{PCOR}$ correspond to robot positions which result in an intersection between the robot and obstacle boundaries. In the worst case, constructing the *C-space* boundary representation requires time $O(m^2)$ and the boundary of *C-space* can have an $O(m^2)$ number of vertices and edges (Latombe[7], Sharir[13]). Although any single connected component of *C-space* was proven to have an $O(m)$ number of vertices, no technique is known to generate them in $O(m)$ time (Sharir[13]). The $\mathcal{PCOR}$ consists of an $O(m)$ edges, each of which may be broken up, in the worst-case, into an $O(m)$ segments, by the $O(m)$ convex obstacles. Thus, in the worst-case, the B-Rep. of the C-space obstacles has an $O(m^2)$ vertices. But, we have shown that the vertices of the $O(m)$ convex obstacles form the reduced search space for solving the *FeasiblePath* problem. Any new vertices in the B-Rep. of the C-space obstacles will be non-convex vertices, and are not required to be visited by the *FeasiblePath* search algorithm. A worst-case example is shown in Figure 5, where the number of C-space vertices

is $O(m^2)$ and the number of $\mathcal{PCOR}$ vertices is $O(m)$.

While an algorithm might spend an $O(m^2)$ time in generating a boundary rep. of *C-space* and then solve the problem of moving a point-robot among *C-space* obstacles having complexity $O(m^2)$, the $\mathcal{WC}$ algorithm (presented below) constructs the $\mathcal{PCOR}$ in $O(m)$ time and solves the problem of moving a point-robot among $\mathcal{PCO}$ obstacles having a complexity of $O(m)$ only. Clearly, the $\mathcal{WC}$ algorithm is working with a less-complex obstacle description, and this is possible because it is found that the precise boundary description of the *C-space* is not required for the solution of the *ShortestPath* or *FeasiblePath* problems in $E^2$.

## 2.5 Window Corner ($\mathcal{WC}$) Algorithm: Two Dimensions

The problem description consists of a robot and a set of obstacles, all of which are arbitrary (non-convex) polygonal objects and are specified as a collection of edges in a planar environment. It is to be determined if there exists some feasible collision-free multistep translation path such that the robot can move from an initial position $S$ to some given goal position $G$, and, if so, to find the shortest feasible path. It is assumed that the start and goal positions are in free space.

We first open the node corresponding to the initial position of the robot and search for new reachable *Window Corners* from the current position of $R$. Every *candidate Window Corner* from $\mathcal{CWC}$ is tested against the conditions for a *Window Corner* from Definition 3 and then it is determined if the *Window Corner* is reachable with a collision-free translation, by the $\mathcal{PCR}$ or $\mathcal{PCOR}$ method. These reachable *Window Corners* are then successively opened until there are no new reachable *Window Corners*. As each reachable *Window Corner* is opened, we also check whether the goal position is reachable from the present position. Alongside, a *Window* ($\mathcal{W}$) *-Graph* is built whose nodes are the reachable *Window Corners* $\cup\{S, G\}$, and whose edges represent the distances between the nodes which each edge connects. ***Actually, only a subset of the convex vertices of the C-space representation is visited.*** A breadth-first search strategy is used to cover the finite search space.

### 2.5.1 $\mathcal{WC} - \mathcal{PCR}$ Algorithm

In this version, we find the set of reachable *Window Corners*, from among $\mathcal{CWC}$ by testing the ***feasiblity*** of each of the corresponding vectors w.r.t. the $\mathcal{PCR}(R, \Phi)$. $\tau(\vec{r})$ denotes the point with position vector $\vec{r}$.

The algorithm proceeds as follows:

1. **Initialize** $\mathcal{CWC}$;
2. **Assign** $\mathcal{CWC} \leftarrow \mathcal{CWC} \cup G$; **Initialize** $OPEN \leftarrow S$;
3. **for each node,** $n \in OPEN$
4. { /*$n$ **is the current position of** $R$ */
5. **Compute** $\mathcal{PCR}$;
6. **for each unreached element,** $c \equiv (v_i^R, v_j^\Phi) \in \mathcal{CWC}$

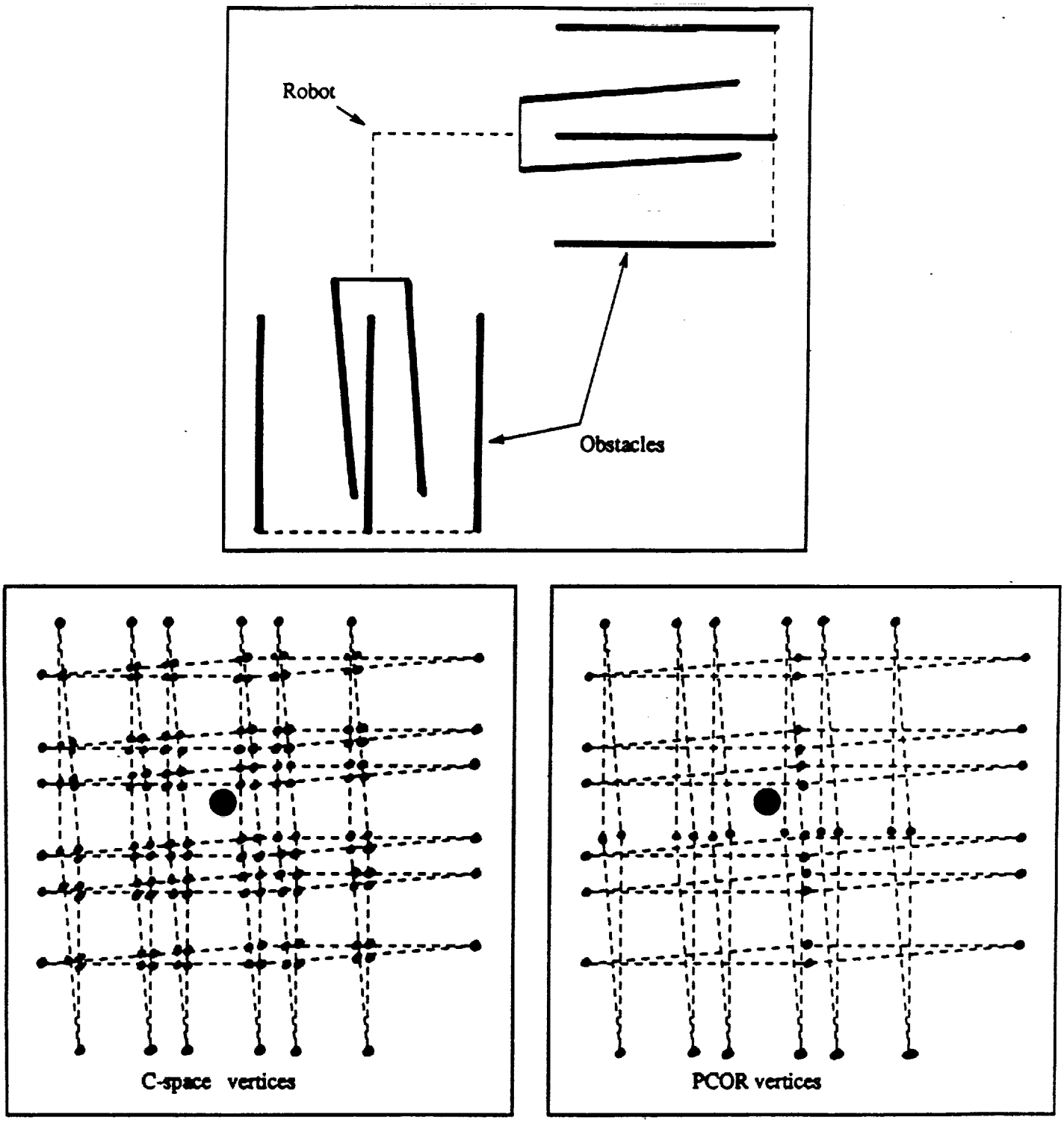


Figure 5: Comparison of $\mathcal{PCOR}$ and a B-Rep. of C-space obstacles

7. {
8. **Let** $\vec{c} = v_j^{\vec{\Phi}} - v_i^{\vec{R}}$;
9. **if** ( $FEAS(\vec{c}, \mathcal{PCR})$ **and** $WC(\tau(\vec{c}), O_{\mathcal{PCR}})$ **and edge** $(n, c) \notin$ *W-Graph*)
10. **then** ($OPEN \leftarrow OPEN \cup \tau(\vec{n} + \vec{c})$; **add edge** $(n, c)$ **to** *W-Graph*)
11. }
12. $OPEN \leftarrow OPEN \setminus n$;
13. }

### 2.5.2 $\mathcal{WC} - \mathcal{PCOR}$ Algorithm

In this version, the set of reachable *Window Corners* w.r.t. the current position are found by using a sweep-line algorithm over the obstacle edge set contained in the $\mathcal{PCOR}$. The search starts from $O_{\mathcal{PCOR}}$ and keeps opening newly reached nodes and adding new edges to the *W-Graph*. The algorithm proceeds as follows:

1. **Compute** $\mathcal{PCOR}$;
2. **Initialize** $OPEN \leftarrow O_{\mathcal{PCOR}}$;
3. **for each node,** $n \in OPEN$
4. { /* $n$ **is the current position of the point-robot** $r$ */
5. **Compute all new positions** $p : \mathcal{WC}(p, n)$;
6. **Let** $OPEN \leftarrow OPEN \cup p$;
7. **Add edge** $(n, p)$ **to** *W-Graph*.
8. $OPEN \leftarrow OPEN \setminus n$;
9. }

### 2.5.3 Discussion

The nodes in the *W-Graph* correspond to the *reachable Window Corners*. Let the number of nodes in the final *W-Graph* = $k$; $1 \leq k \leq (m + 2)$. Hence, the *W-Graph* can consist of $O(k^2)$ edges.

**Theorem 4** *The* $\mathcal{WC} - \mathcal{PCR}$ *algorithm constructs the W-Graph in time* $O(km^2) \leq O(m^3)$.

**Proof:** At each reachable position, an $O(m)$ vectors of the $\mathcal{CWC}$ are tested for feasibility with respect to the $\mathcal{PCR}$. Since the $\mathcal{PCR}$ consists of an $O(m)$ constraint cones, the total test at each node takes $O(m^2)$ time. Since $k$ nodes are opened, the entire algorithm takes $O(km^2)$ time to construct the *W-Graph*, where, in the worst case, $k = O(m)$.

**Q.E.D.** □

The *FeasiblePath* problem is solved if $G$ is reachable from some *Window Corner* reachable from $S$. The W-Graph can be searched for the shortest path between $S$ and $G$, giving the solution to the *ShortestPath* problem.

**Theorem 5** *The $\mathcal{WC} - \mathcal{PCOR}$ algorithm constructs the* W-Graph *in time* $O(km\log(m)) \leq O(m^2\log(m))$.

**Proof:** At each node, the sweep-line algorithm takes $O(m\log(m))$ time to identify the new reachable *Window Corners*. This is because, the obstacle set in the $\mathcal{PCOR}$ consists of an $O(m)$ edges. Since $k$ nodes are opened, the $\mathcal{WC} - \mathcal{PCOR}$ algorithm takes $O(km\log(m))$ time.

**Q.E.D.** □

**Theorem 6** *Using the $\mathcal{WC} - \mathcal{PCOR}$ algorithm, the FeasiblePath and ShortestPath problems can be solved in $O(km\log(m)) \leq O(m^2\log(m))$ time.*

**Proof:** Once the *W-Graph* is built, it can be searched for the shortest path from $S$ to $G$. Using Dijkstra's algorithm, the shortest path among $k$ nodes in the *W-Graph* can be found in $O(k^2)$ time. As a result of Theorem 5, the proof follows.

**Q.E.D.** □

*The nodes in the W-Graph are a subset (Window Corners) of the convex vertices of the C-space representation of the problem.* Let $n$ be the number of vertices in the *C-space* representation of the problem. $O(n^2)$ time algorithms are known (Welzl[14], Asano et al [1]) to solve the *C-space* representation of the problem, but these algorithms have certain assumptions such as non-intersecting edges (or polygons) in the input. The previously fastest algorithm was an $O(n^2\log(n))$ algorithm using a sweep-line technique (Sharir et al [12]). The worst-case bound on the number of *C-space* vertices is $n = O(m^2)$ (Latombe[7], Sharir[13]). Since it does not construct *C-space*, the $\mathcal{WC} - \mathcal{PCOR}$ algorithm is expected to be faster than any method which first constructs the *C-space*. Further, any *C-space* based algorithm is applicable to the $\mathcal{PCOR}$, since the *C-space* can be computed from the $\mathcal{PCOR}$ in $O(m^2)$ time.

### 2.6 Example

We tested the $\mathcal{WC}$ algorithm on an assembly planning problem. The objective was to determine whether a feasible collision-free multistep translation path existed to separate a non-convex polygonal part situated in a maze-type assembly, consisting of some tightly constrained regions. The $\mathcal{WC}$ algorithm successfully solved the *FeasiblePath* problem, and reported a feasible (and shortest) path consisting of 27 translations, in under 100ms. This result is shown in figure 6.

## 3 The FeasiblePath problem in Three Dimensions

### 3.1 Preliminaries

We introduce the following definitions:

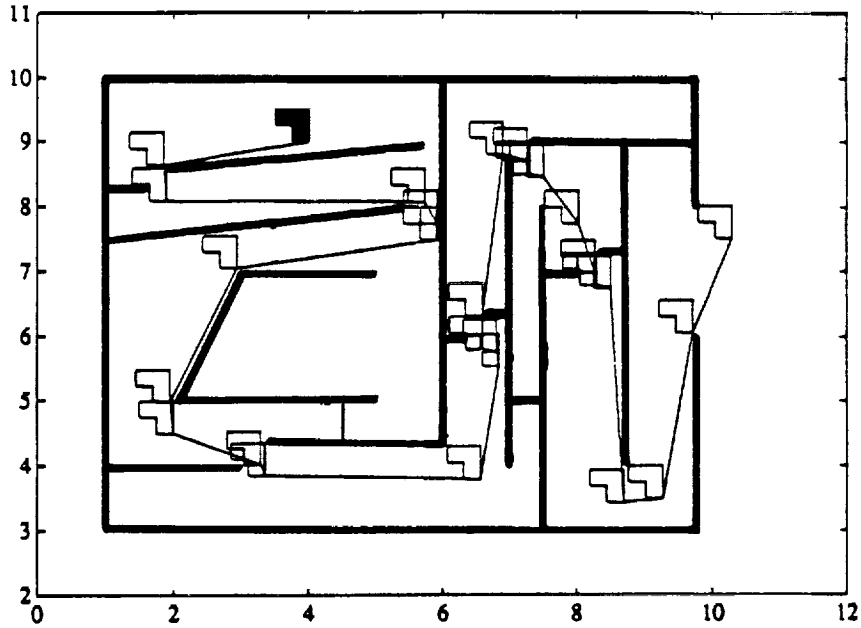


Figure 6: A *FeasiblePath* problem and its solution using the *Window Corner* algorithm

- $\pi_V \triangleq$ a set of vertices, $\{v_i^\pi\}\ \forall\ i = 1, \ldots, v(\pi)$,
- $\pi_E \triangleq$ a set of edges, $\{e_i^\pi\}\ \forall\ i = 1, \ldots, e(\pi)$, where the end points of each edge are straight-line reachable from each other, ( without loss of clarity, we will sometimes refer to $\pi_E$ as the set of all points contained in these edges ), $\pi_V \subset \pi_E$,
- $\pi_F \triangleq$ a set of faces, $\{f_i^\pi\}\ \forall\ i = 1, \ldots, f(\pi)$, ( we will sometimes refer to $\pi_F$ as the set of all points contained in these faces ), $\pi_E \subset \pi_F$,
- $\pi_R \triangleq$ set of (or the points in the) specified polyhedral regions, that the faces may enclose,
- $\pi \triangleq \{x : x \in E^3 \wedge (x \in \pi_F \vee x \in \pi_R)\}$, the set of all points contained in the polyhedral regions, faces, edges and vertices,
- $F \triangleq E^3 \setminus \pi$ (the free space).

In this paper, a point $q$ is said to be *straight-line reachable* from another point $p$, if the straight-line translation from $p$ to $q$ is collision-free.

Given two points $p, q \in E^3$, $\vec{q}$ denotes the position vector corresponding to the point $q$, $\vec{pq} = \vec{q} - \vec{p}$, $\overline{pq}$ denotes the (interchangeably, the set of points in) line segment between $p$ and $q$, and $\|\overline{pq}\|$ denotes the length of the line segment from $p$ to $q$. We now introduce the following definitions. Let $p \in F$.

**Definition 16** *An $\alpha$-neighborhood of $p$ is defined as a set $\mathcal{N}_\alpha(p) = \{u : u \in E^3, \|\overline{pu}\| \leq \alpha\}$ where $\alpha$ is a small, positive, real number.*

**Definition 17** *A $\beta$-neighborhood of $p$ is defined as a set $\mathcal{N}_\beta(p) = \{u : u \in \pi, \|\overline{pu}\| \leq \beta\}$ where $\beta$ is a small, positive, real number.*

**Definition 18** *An $\epsilon$-neighborhood of an edge $e$, with vertices $v_1$ and $v_2$, is defined as a set $\mathcal{N}_\epsilon(e) = \{u : u \in \pi, w \in e, \|\overline{wu}\| \leq \epsilon, \vec{wu} \cdot \vec{v_1 v_2} = 0\}$ where $\epsilon$ is a small, positive, real number.*

**Definition 19** *An edge $e \in \pi_E$ is a* **Window Edge** *w.r.t. $p$, written as $\mathcal{WE}(e, p)$, iff, there exists a small $\epsilon$ and a normal, $\vec{n}$, to the plane containing $e$ and $p$, such that $\vec{n} \cdot \vec{pv} \geq 0 \quad \forall\ v \in \mathcal{N}_\epsilon(e)$. $\mathcal{WP}(x, p)$ states that point $x$ is located on an edge which is a* Window Edge *w.r.t. $p$.*

## 3.2 The Window Edge and Feasible Path Theorems

**Definition 20** *A collision-free path, $\Psi$, from some initial position $S$ to some goal position $G$, is defined as a sequence of $n$ points $\psi_1 \psi_2, \ldots, \psi_n$ such that $\overline{\psi_i \psi_{i+1}} \cap int(\pi) = \emptyset, \forall\ i = 1, \ldots, (n-1)$, and where $\psi_1$ corresponds to $S$ and $\psi_n$ corresponds to $G$.*

Assume that we are given two points $p, q \in F$, but $\overline{pq}$ is not collision-free i.e. $\overline{pq} \cap \pi \neq \emptyset$. Also assume that the shortest collision-free path from $p$ to $q$ is $p\omega q$, where $\omega \in E^3$, $\overline{p\omega} \cap \pi = \emptyset$, $\overline{\omega q} \cap \pi = \emptyset$. We now state the following theorem:

**Theorem 7 (Window Edge Theorem)** *If the shortest collision-free path from $p$ to $q$ changes direction at a single point $\omega$, then $\omega$ lies on a Window Edge w.r.t. $p$.*

**Proof:** The proof is presented in two parts.

1. $\omega \in \pi_E$: By contradiction. Let $((a \in (\mathcal{N}_\alpha(\omega)) \cap \overline{p\omega}) : (\|\overline{a\omega}\| = \alpha))$ and $((b \in (\mathcal{N}_\alpha(\omega)) \cap \overline{\omega q}) : (\|\overline{\omega b}\| = \alpha))$. If $(\omega \in F)$, see Figure 7(a), then $\exists \alpha : \mathcal{N}_\alpha(\omega) \cap \pi = \emptyset$, hence $\overline{ab} \cap \pi = \emptyset$. If $((\omega \in \pi_F) \wedge (\omega \notin \pi_E))$, see Figure 7(b) then $\exists \alpha : \overline{ab} \cap \pi = \emptyset$. But, $\|\overline{ab}\| < \|\overline{a\omega}\| + \|\overline{\omega b}\|$. Hence, $\|\overline{pa}\| + \|\overline{ab}\| + \|\overline{bq}\| < \|\overline{p\omega}\| + \|\overline{\omega q}\|$. Thus, if $\omega \notin \pi_E$, a shorter path can be found and so $\omega$ has to lie on an obstacle edge.

2. Let $\omega$ lie on an obstacle edge $e$. We now show, by contradiction, that there should exist a normal $\vec{n}$, to the plane containing $e$ and $p$, for some small, positive $\epsilon$, such that

$$\vec{n} \cdot \vec{pv} \geq 0 \quad \forall\ v \in \mathcal{N}_\epsilon(e). \tag{9}$$

   A situation to the contrary is shown in Figure 7(c). In such a case, there exist some two points $a$ and $b$, as defined above, for some small, positive, real $\alpha$, such that $a, b \in \alpha(\omega)$ and $\|\overline{pa}\| + \|\overline{ab}\| + \|\overline{bq}\| < \|\overline{p\omega}\| + \|\overline{\omega q}\|$. Hence, a shorter path can be found and so $\omega$ should be such as to satisfy the condition (1) above.

The above (1) and (2) are necessary and sufficient conditions for $\omega$ to lie on a *Window Edge* w.r.t. $p$, i.e. $\mathcal{WP}(\omega, p)$. from Definition 19.

( In fact, the following condition should also be satisfied; $\vec{n} \cdot \vec{\omega q} > 0$.)

**Q.E.D.** □

**Corollary 2** *Given the shortest collision-free path, $\Psi$, as in Definition 20, every intermediate point on this path lies on a Window Edge with respect to it's predecessor on the path, i.e. $\mathcal{WP}(\psi_{i+1}, \psi_i)\ \forall\ i = 1, \ldots, (n-2)$ and $\mathcal{WP}(\psi_i, \psi_{i+1})\ \forall\ i = 2, \ldots, (n-1)$.*

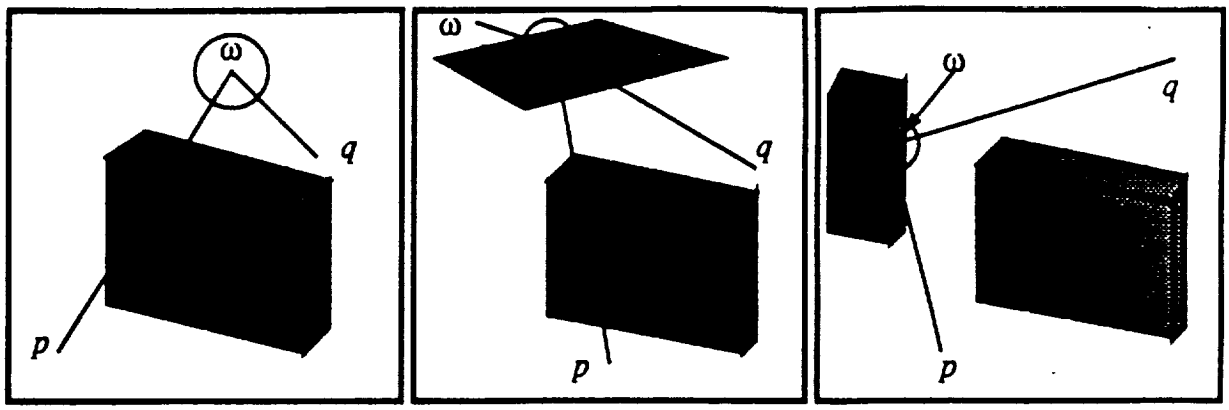


Figure 7: Proof of the Window Edge Theorem

**Proof:** For $\Psi$ to be the shortest collision-free path from $\psi_1$ to $\psi_n$, it is necessary that it represents the shortest collision-free path between any three successive points in $\Psi$. In other words, $\psi_i\psi_{i+1}\psi_{i+2}$ is the shortest collision-free path between $\psi_i$ and $\psi_{i+2}$, $\forall\, i = 1, \ldots, (n-2)$. From the **Window Edge Theorem**, it follows that $\mathcal{WP}(\psi_{i+1}, \psi_i)\ \forall\ i = 1, \ldots, (n-2)$ and $\mathcal{WP}(\psi_i, \psi_{i+1})\ \forall\ i = 2, \ldots, (n-1)$.

**Q.E.D.** □

Although the successor point on a shortest path lies on a *Window Edge* w.r.t. its predecessor, the problem of locating the actual points on the *Window Edges* requires exhaustive calculations, since a continuous space needs to be searched. It is for this reason that the problem of computing the shortest collision-free path in three- dimensions has been found to be exponentially difficult (Sharir et al[12], Papadimitrou [11]). Interestingly, for finding a feasible path, it is sufficient to visit a *Window Edge* once. The precise point on the *Window Edge*, that is visited, is not critical to finding a feasible path, if one exists. This is shown by the following corollary and a theorem. Let there exist a shortest collision-free path $\Psi$, as in Definition 20, from some point $p$ to $G$. Let the first intermediate point on this be $\psi_1 \in \mathcal{W}_i$ where $\mathcal{W}_i$ is a Window Edge w.r.t. $p$.

**Corollary 3** *Every point $x \in \mathcal{W}_i$, which is straight-line reachable from $p$, lies on some feasible path from $p$.*

**Proof:** For any point $x \in \mathcal{W}_i$, an obvious feasible path from $p$ to $G$ exists. Such a path first visits $x$ and then visits the intermediate points of $\Psi$, begining from $\psi_1$, in sequence. This is because $\psi_1$ is straight-line reachable from every point $x \in \mathcal{W}_i$.

**Q.E.D.** □

Let some present position be $p$. Let the set of Window Edges w.r.t. $p$ be $\mathcal{W}_i$. Let $\mathcal{Y}$ be a set of points, defined as consisting of exactly one straight-line reachable (from $p$) point (if one exists), from each $\mathcal{W}_i$.

**Theorem 8 (Feasible Path Theorem)** *There exists a feasible path from $p$ to $G$, if and only if, there exists a feasible path through at least one of the points in $\mathcal{Y}$.*

**Proof:** The first part is obvious. For the second part, assume the contrary. Let there exist a feasible path from $p$ to $G$ but that none of the points in $\mathcal{Y}$ lies on a feasible path. Obviously, there exists a shortest path from $p$ to $G$. The first intermediate point $\psi_1$ will clearly lie on some *Window Edge* $\mathcal{W}_j$. From Corollary 2, there exists a feasible path through every reachable point on $\mathcal{W}_j$. Since $\mathcal{Y}$ consists of one such point from $\mathcal{W}_j$, there exists a feasible path from $p$ to $G$ through one of the points in $\mathcal{Y}$. We have arrived at a contradiction and hence the proof.

**Q.E.D.** □

As a result of the **Feasible Path Theorem**, we can proceed with the search as follows. Starting from the initial position, we keep visiting some point on a new *Window Edge* until either the goal position is reached or there are no new *Window Edges* to visit. Proceeding in this way, a *Window Edge Tree* can be built which represents the various sequences of *Window Edges* that have been visited.

**Theorem 9** *To be able to detect a feasible path from p to G, each edge $e_i^\pi$ needs to be visited in at most one point, resulting in a finite search space of size $= e(\pi)$.*

**Proof:** Let edge $e_i^\pi$ be visited at point $y$. It is seen that there exists a feasible path passing through some point $x \in e_i^\pi, x \neq y$, if and only if there exists a feasible path passing through $y$. Hence, the existence of a feasible path is guaranteed to be detected, even if at most one point $y$ on every edge $e_i^\pi$, is expanded. Since the maximum number of edges that can be visited, is $e(\pi)$, the proof follows.

**Q.E.D.** □

We now introduce *Window Corners* which identify reachable locations, on *Window Edges*. The *Window Corner* algorithm will trace a path through *Window Corners* for solving the *FeasiblePath* problem.

**Definition 21** *If a ray emanating from position p, intersects $\Phi$ such that the first n points of intersection lie on Window Edges, $\mathcal{W}_i$ w.r.t. p, then these intersection points are called Window Corners corresponding to the Window Edge $\mathcal{W}_i$ $\forall i = 2, \ldots, n$.*

## 3.3 Polyhedral Cone Representation ($\mathcal{PCR}$)

For the rest of the paper, we consider the robot, $R$, and obstacles, $\Phi$, to be described as follows: The robot has $v(R)$, $e(R)$ and $f(R)$ vertices ($v_i^R$), edges ($e_i^R$), and faces ($f_i^R$) respectively. $R$ also denotes the set of all points contained in the robot. The obstacles have $v(\Phi)$, $e(\Phi)$ and $f(\Phi)$ vertices ($v_i^\Phi$), edges ($e_i^\Phi$), and faces ($f_i^\Phi$) respectively. $\Phi$ also denotes all points contained in the obstacles. Note that the robot and obstacles are basically a collection of faces in a polyhedral environment. We assume that each subassembly is equal to the closure of its interior points and that each face is an orientable surface. In addition, $S$ and $G$ refer to the start and goal positions of some point on the robot, respectively. Also, we define the space complexity of certain terms as under, where $m = v(R)v(\Phi)$:

$$e(R)e(\Phi) = O(m), \quad f(R)f(\Phi) = O(m), \quad v(R)f(\Phi) = O(m), \quad f(R)v(\Phi) = O(m). \tag{10}$$

The constraint on the translation of a vertex with respect to a convex, planar face can be represented as a *Vertex-Face (VF) constraint cone*, which can be defined as follows.

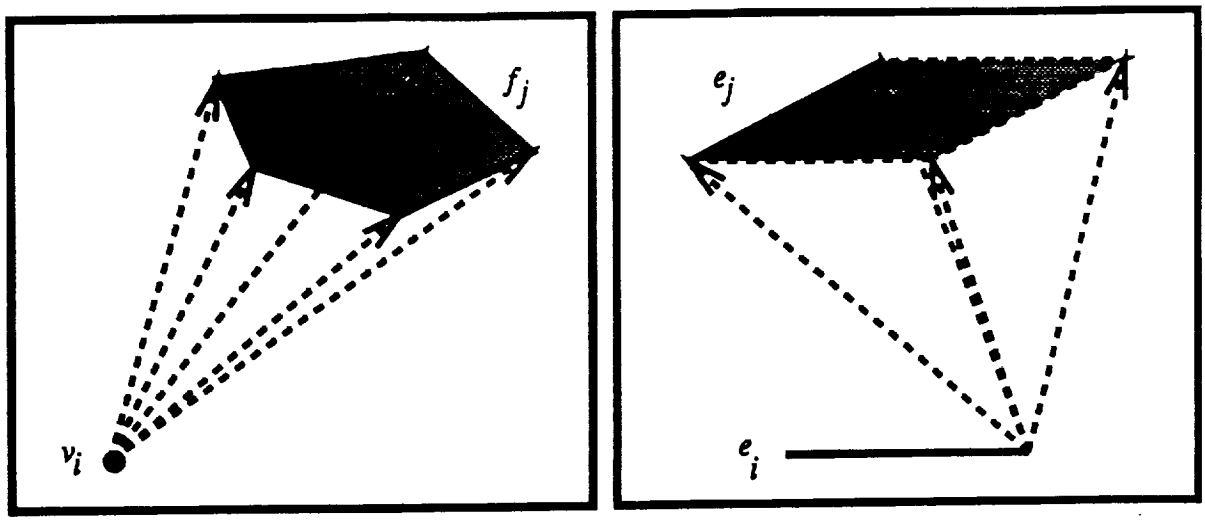


Figure 8: (a) *Vertex-Face* and (b) *Edge-Edge* constraint cones

**Definition 22** *A Vertex-Face constraint cone $C_{VF}(v_i, f_j)$, between a vertex, $v_i$, and a face, $f_j$ (with $v(f_j)$ vertices), consists of bounding vectors, $\vec{s_k}$ $\forall k = 1, \ldots, v(f_j)$, drawn from the vertex to the corners of the face, and a base, which is the face itself. A VF cone, consisting of an apex, a base and the bounding vectors, is shown in Figure 8(a).*

The constraint on the translation of an edge with respect to another edge can be represented as an *Edge-Edge (EE) constraint cone*, which can be defined as follows.

**Definition 23** *An Edge-Edge constraint cone, $C_{EE}(e_i, e_j)$, between an ordered pair of edges, $e_i$ and $e_j$, consists of four bounding vectors, $\vec{s_1}$, $\vec{s_2}$, $\vec{s_3}$ and $\vec{s_4}$, drawn from the vertices of $e_i$ to the vertices of the $e_j$, and the planar base formed by the end-points of bounding vectors. An EE cone, consisting of an apex, a base and four bounding vectors, is shown in Figure 8(b).*

A *VF cone* or an *EE cone* can be unambiguously represented by unit vectors $\vec{n_{s_k}}, \vec{n_B}$, (which are the normals to the bounding faces and the base of the cone respectively) and one of the bounding vectors, say, $\vec{s_1}$. These vectors are collectively called the *normal representation* of the *VF cone*.

**Definition 24** *The normal representation of the negation of a VF (or EE) cone, denoted by*
*$-C_{VF}(v_i, f_j)$ (or $-C_{EE}(e_i, e_j)$), is obtained by reversing the direction of each of the vectors in the normal representation of the VF (or EE) cone.*

We are now ready to define the Polyhedral Cone Representation, $\mathcal{PCR}$, as follows. Given a robot, $R$, at some position and orientation in $E^3$, and a set of obstacles, $\Phi$:

**Definition 25** *The set of all (fully three-dimensional i.e. non-degenerate) VF and EE cones between the robot and obstacles, is called the Polyhedral Cone Representation, $\mathcal{PCR}(R, \Phi)$, or simply $\mathcal{PCR}$, i.e.*

$$\mathcal{PCR} = \{C_{VF}(v_i^R, f_j^\Phi) \forall\ i,j\} \cup \{-C_{VF}(v_k^\Phi, f_l^R) \forall\ k,l\} \cup \{C_{EE}(e_x^R, e_y^\Phi) \forall\ x,y\}. \quad (11)$$

The origin of the $\mathcal{PCR}$ is denoted by $O_{\mathcal{PCR}}$.

**Definition 26** *A vector $\vec{v} \in R^3$ is said to be infeasible w.r.t. some VF or EE cone, $C_j$ (with $b$ bounding vectors), written $INFEAS(\vec{v}, C_j)$, iff:*

1. $\vec{v} \cdot \vec{n_{s_k}} \geq 0, \quad \forall k = 1, \ldots, b$ *and*
2. $\vec{n_B} \cdot (\vec{v} - \vec{s_1}) \leq 0.$

*If semi-free paths are allowed, conditions 1 and 2 are strict inequalities. In the case of an unbounded cone, only condition 1 holds.*

**Definition 27** *Vector $\vec{v}$ is infeasible w.r.t. the $\mathcal{PCR}$, i.e.*

$$INFEAS(\vec{v}, \mathcal{PCR}) \Leftarrow if\ \exists C_j : (C_j \in \mathcal{PCR}) \wedge (INFEAS(\vec{v}, C_j)). \quad (12)$$

*A vector that is not infeasible w.r.t the $\mathcal{PCR}$, is said to be feasible w.r.t. the $\mathcal{PCR}$, $FEAS(\vec{v}, \mathcal{PCR})$.*

**Theorem 10** *A translation of the robot by some vector $\vec{v}$, from it's current position is guaranteed to be collision-free iff $FEAS(\vec{v}, \mathcal{PCR})$.*

**Proof:** We know from Definitions 26 and 27 that if $FEAS(\vec{v}, \mathcal{PCR})$, then $\vec{v}$ does not pierce the base of any constraint cone. From Definitions 22 and 23, this implies that no robot (obstacle) vertex collides with an obstacle (robot) face, and that there are no collisions between robot and obstacle edges. These are necessary and sufficient conditions for any translation among polyhedral objects to be collision-free, and hence the proof.

**Q.E.D.** □

Any non-convex face of the robot or obstacle is first decomposed into convex parts, which are then used for the computation of the $\mathcal{PCR}$, resulting in $O(m)$ constraint cones.

A vector is infeasible w.r.t. the $\mathcal{PCR}$ iff it pierces the base of any *VF or EE cone*. Hence, for the motion of $O_{\mathcal{PCR}}$, we can consider the bases of the *VF and EE cones* to be obstacle faces. We know from Definition 19 that *Window Edges* correspond to the edges of the obstacle faces. Therefore, the potential *Window Edges* with respect to $O_{\mathcal{PCR}}$, are the edges of the *VF and EE cone* bases.

In fact, each such base edge is also the base edge of some *EE cone*. An inspection of the *EE cones* from Definition 23 shows that these base edges correspond to a contact between some robot (obstacle) vertex and an obstacle (robot) edge.

Now we are in a position to identify the *Window Corners* w.r.t. $O_{\mathcal{PCR}}$. We visit a *Window Edge* at one of it's vertices if the corresponding translation vector is feasible w.r.t. the $\mathcal{PCR}$. If both the vertices are unreachable, we compute the point on the *Window Edge*, which lies in the direction of the line of intersection of the triangular facets supported by

the *Window Edge* and some other *Window Edge*. These facets are supported at $O_{\mathcal{PCR}}$ and are the bounding face of the *EE cone* corresponding to the *Window Edges*.

The points on the *Window Edges* which are visited, using the above technique, are the previously defined *Window Corners*. The search space to find the *Window Corners* w.r.t. $O_{\mathcal{PCR}}$, is $O(m^2)$, since there are an $O(m)$ *Window Edges*.

### 3.4 Polyhedral Cone Obstacle Representation ($\mathcal{PCOR}$)

Given two convex planar faces, $f_1$ and $f_2$, it is observed that the base-faces of all the *VF and EE cones* between the two faces, enclose a convex polyhedron whose interior is entirely unreachable by a collision-free translation from $O_{\mathcal{PCR}}$.

**Definition 28** *A $\mathcal{PCO}$ between a pair of convex faces is a convex polyhedron whose bounding faces correspond to the base-faces of some* VF or EE cone *between the two faces. We will sometimes refer to a $\mathcal{PCO}$ as the set of all points contained in the polyhedron.*

**Definition 29** *The set of all $\mathcal{PCO}$s computed between robot faces and obstacle faces , at $S$, is called the Polyhedral Cone Obstacle Representation ($\mathcal{PCOR}$) of the problem:*

$$\mathcal{PCOR} = \{\mathcal{PCO}(i,j)\} \quad \forall\, i = 1,\ldots,f(R) \quad \forall\, j = 1,\ldots,f(\Phi) \tag{13}$$

*We also refer to $\mathcal{PCOR}$ as the set of all points contained in the constituent $\mathcal{PCO}$s.*

**Definition 30** *A vector $\vec{v} \in R^3$, is feasible w.r.t. a $\mathcal{PCO}$ iff: $\vec{v} \cap int(\mathcal{PCO}) = \emptyset$. This can be determined by testing the vector against each of the faces comprising the $\mathcal{PCO}$. A vector that is feasible w.r.t every $\mathcal{PCO}$, is said to be feasible w.r.t. the $\mathcal{PCOR}$; $FEAS(\vec{v}, \mathcal{PCOR})$.*

It follows from the above that:

$$if\ \exists \mathcal{PCO}(i,j) : v \in int(\mathcal{PCO}(i,j)) \Rightarrow INFEAS(\vec{v}, \mathcal{PCR}) \tag{14}$$

Let the origin of the $\mathcal{PCOR}$ be $O_{\mathcal{PCOR}}$; it corresponds to the start position of the robot. In the $\mathcal{PCOR}$, the various $\mathcal{PCO}$s form the obstacles for the motion of a point, henceforth called the point-robot $r$, from $O_{\mathcal{PCR}}$ to a point whose position vector is $\vec{G} - \vec{S}$.

**Theorem 11** *Finding a feasible (or shortest) collision-free path, $\Psi$, for the point-robot from position $O_{\mathcal{PCR}}$ to position $\vec{G} - \vec{S}$, in the $\mathcal{PCOR}$, such that $FEAS(\vec{\psi_{i+1}} - \vec{\psi_i}, \mathcal{PCOR}) = \emptyset, \forall\, i = 1,\ldots,(n-1)$, is also the solution to the original FeasiblePath (or ShortestPath) problem.*

**Proof:** It is easily verified that the boundary faces of the $\mathcal{PCO}$s are the bases of the *VF and EE cones* between robot and obstacle faces. Avoiding the interior of each $\mathcal{PCO}$ is equivalent to avoiding a non-contact collison between all robot and obstacle faces, since all edge-face collisions are avoided, from (14). The goal position for the point-robot is simply found to be $\vec{G} - \vec{S}$. $\Psi$, therefore provides the relative coordinates for the locus of some point on the robot $R$, and hence the solution to the original problem.

**Q.E.D.** □

### 3.4.1 $\mathcal{PCOR}$ and *C-space*

Hence, in the $\mathcal{PCOR}$, the problem is to find a collision-free path for a point-robot moving from the start position,$O_{\mathcal{PCOR}}$, to a goal position, $\vec{G} - \vec{S}$, among a set, $O(m)$, of convex polyhedral obstacles ($\mathcal{PCO}s$). The $\mathcal{PCOR}$ is a convex constructive representation (superset) of the *C-space* obstacle boundaries. Collectively, the $\mathcal{PCOR}$ corresponds to robot positions which result in an intersection between the robot and obstacle boundaries. In the worst case, constructing a boundary rep. of the *C-space* requires time $O(m^3)$, and the boundary of *C-space* can have an $O(m^3)$ number of vertices and edges (Latombe[7], Sharir[13]). There are an $O(m)$ faces in the $\mathcal{PCOR}$. Each face is broken into sub-faces, due to intersections with the $O(m)$ convex obstacles in the $\mathcal{PCOR}$. In the worst case, each face may be divided into an $O(m^2)$ unconnected sub-faces. The boundary of *C-space* is the union of a subset of these sub-faces, resulting in an $O(m^3)$ worst-case space complexity for the boundary representation. In regions where motion is completely constrained by contacts, the feasible translations may lie on surfaces with no interior points. Such surfaces can be inaccessible in a typical *C-space* boundary representation. *Such contact surfaces are accessible in our $\mathcal{PCOR}$ representation, since all face-face constraints are individually represented.*

Therefore, while an algorithm might spend an $O(m^3)$ time in generating *C-space* and then solve the feasibility problem of moving a point-robot among *C-space* obstacles having complexity $O(m^3)$, we have shown that an algorithm could construct the $\mathcal{PCOR}$ in $O(m)$ time and solve the feasibility problem of moving a point-robot among $\mathcal{PCO}$ obstacles having a complexity of $O(m)$ only. The $\mathcal{WC}$ algorithm is working with a less-complex obstacle description, and this is possible because it is found that the precise boundary description of the *C-space* is not required for the solution of the *ShortestPath* or *FeasiblePath* problems in $E^3$. In addition, the *Window Edges* correspond to a subset of the convex edges of the boundary representation of *C-space*.

## 3.5 Hierarchical Assembly Path Planning

The problem environment consists of a moving-subassembly (robot) and a collection of obstacle-subassemblies, all of which are assumed to be polyhedral, and are described by a set of faces. We assume that each subassembly is equal to the closure of its interior points and that each face is an orientable surface. The initial and goal positions for the robot are also specified. In assembly sequence planning, a goal configuration corresponding to a disassembly can be set up trivially. The *Window Corner* algorithm determines if there exists some collision-free path for the robot to reach the goal configuration, and enumerates such a path. We assume that the robot is not completely contained within any obstacle.

### 3.5.1 Feasibility of Incremental Translation

We now determine the feasibility of incremental translational motion, based on analyzing all the (not only planar) contact constraints. We first compute the $\mathcal{PCOR}$. If the $\mathcal{O}_{\mathcal{PCOR}}$ lies in the interior of at least one $\mathcal{PCO}$, then the robot is intersecting the obstacles, and therefore no feasible incremental motion is possible.

If $\mathcal{O}_{\mathcal{PCOR}}$ lies on the boundary of $\mathcal{PCO}_j$, then this contact gives rise to an *unbounded, convex Local Constraint Cone*, $LC_j$, whose normal representation consists of the normal

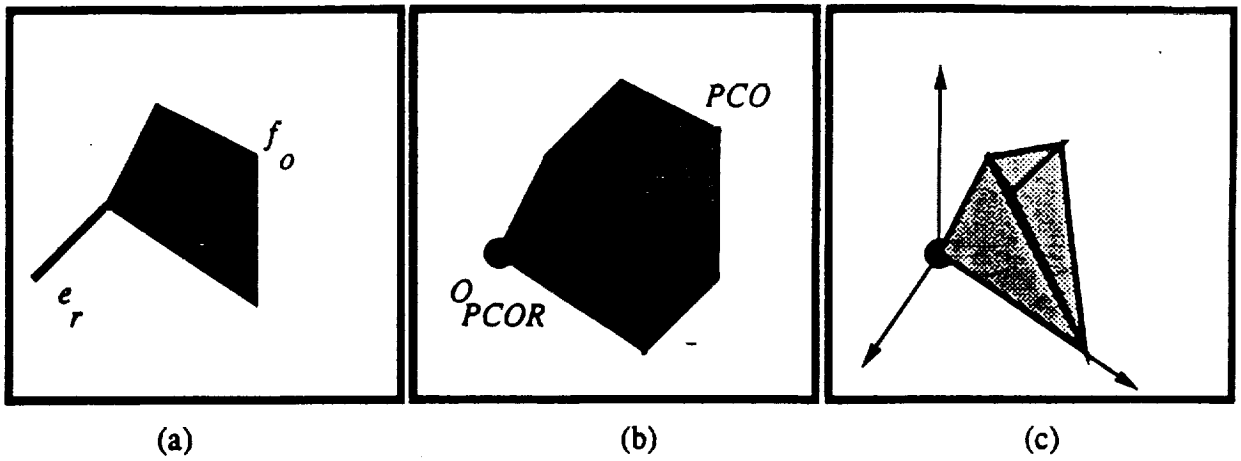


Figure 9: Computing *Local Constraint Cones*

(in the direction of $\mathcal{PCO}_j$) to the $\mathcal{PCO}$ face on which $\mathcal{O}_{\mathcal{PCOR}}$ lies. Note that the normal representation of $LC_j$ has 3, 2 or 1, vectors depending on whether $\mathcal{O}_{\mathcal{PCOR}}$ lies on a vertex, edge, or the interior of a face of $\mathcal{PCO}_j$ respectively.

The set of all $LC_j$ for all (say, $k$ in number) the $\mathcal{PCO}_j$, in contact with $\mathcal{O}_{\mathcal{PCOR}}$ is called the *Local Cone Representation* or $\mathcal{LCR}$.

Clearly, any incremental motion is possible if and only if, there exists some unbounded vector (a ray) which is feasible w.r.t. every contacting $\mathcal{PCO} \in \mathcal{PCOR}$. Using the preceding discussion on *Window Corners*, it is seen that there exists such a feasible vector if and only if one of the rays containing a *Window Corner* is feasible. Hence, the possible unbounded vectors are the individual bounding vectors of the local constraint cones or the vectors corresponding to the intersection of the boundary of a pair of local constraint cones. If the number of contacts is small, then the feasibility of incremental motion can be determined efficiently (in $O(k^3)$ time), before proceeding with the more computationally expensive search for multiple-step paths.

Figure 9(a) shows an obstacle face and a robot edge. The $\mathcal{PCO}$ is given in Figure 9(b). Since $\mathcal{O}_{\mathcal{PCOR}}$ lies on a vertex of the $\mathcal{PCO}$, the *Local Constraint Cone* is the intersection of three half-spaces, Figure 9(c).

### 3.5.2 Testing for Straight-line Assembly Paths

In this subsection, we show how to determine the existence of a straight-line path to separate a pair of non-convex polyhedral subassemblies. It is to be determined if there exists some unbounded vector (ray) which is feasible w.r.t. the $\mathcal{PCOR}$. The procedure followed is similar to that in the previous section and requires obtaining rays, containing *Window Corners*, which are feasible w.r.t. $\mathcal{PCOR}$. Hence, determining the existence of a straight-line path to separate a pair of subassemblies requires $O(m^3)$ time.

### 3.5.3 Paths with Multiple-Translations

If some incremental translation is feasible, and a straight-line assembly path does not exist, the algorithm searches for a path with multiple translations. We first open the node corresponding to the initial position of the robot and identify the *Window Edges* w.r.t. the present position. A convex decomposition of non-convex faces is carried out, prior to the construction of the $\mathcal{PCOR}$. We then identify the *Window Corners* on these *Window Edges*, using the technique described in Subsection 3.3. It is then determined if the *Window Corner* is reachable with a collision-free translation, w.r.t. the $\mathcal{PCOR}$. These reachable *Window Corners* are then successively opened until there are no reachable *Window Corners*, which are unreachable from any previously visited *Window Corners* on the same *Window Edge* (or until the goal position is reached). As each reachable *Window Corner* is opened, we also check whether the goal position is reachable from the present position. Alongside, a *Window Edge*($\mathcal{WE}$) *-Tree* is built whose nodes are the reachable segments on *Window Edges* $\cup\{S, G\}$, and whose edges denote the traversability between the *Window Edges* which are connected. Thus, a breadth-first search ripples out, to cover the finite search space. *From a C-space perspective, the WC algorithm restricts the search process to only a subset of the convex edges of the corresponding C-space representation.*

### 3.5.4 3D *WC* Algorithm for Feasible Paths

1. **Initialize** $OPEN \leftarrow O_{PCOR}$; $GOAL = FALSE$;
2. **do** {
3. **for each node,** $n \in OPEN$
4. { /***n is the current position of** $R$ **on some** *Window Edge* **segment** $e_1$ */
5. **Construct the** $\mathcal{PCOR}$;
6. **if** $FEAS((\vec{G} - \vec{S} - \vec{n}), \mathcal{PCOR})$ **then GOAL = TRUE;**
7. **Identify the** *Window Edges and Window Corners*;
8. **for each** *Window Corner* $c$, **on some** *Window Edge* **segment** $e_2$,
9. {
10. **if** $((FEAS(\vec{c} - \vec{n}, \mathcal{PCOR})$ **and edge** $(e_1, e_2) \notin$ $\mathcal{WE}$*-Tree*$))$
11. **then** $\{(OPEN \leftarrow OPEN \cup c)$; **add edge** $(e_1, e_2)$ **to** $\mathcal{WE}$*-Tree;*$\}$
12. }
13. $OPEN \leftarrow OPEN \setminus n$;
14. }
15. } **while (GOAL == FALSE);**

### 3.5.5 Discussion

The nodes in the $\mathcal{WE}$-*Tree* correspond to the reachable *Window Corners* on *Window Edges*. Let the number of nodes in the final $\mathcal{WE}$-*Tree* $= k$; $1 \leq k \leq u : u = O(m^2)$. Hence, the $\mathcal{WE}$-*Tree* can consist of $O(m^2)$ edges, since each reachable segment is only visited once. The $n$ arbitrary faces of a polyhedron with $O(n)$ edges, can be decomposed into an $O(n)$ convex faces since each convex face could be bounded by at least one of the original polyhedron's edges. Hence, the number of constraint cones in the $\mathcal{PCR}$, is $O(m)$.

**Theorem 12** *The WC algorithm constructs the* $\mathcal{WE}$*-Tree and solves the FeasiblePath problem in time* $O(m^5)$.

**Proof:** At each reachable position, an $O(m^2)$ vectors of the *Window Corners*, are tested for feasibility with respect to the $\mathcal{PCR}$. Since the $\mathcal{PCR}$ consists of an $O(m)$ $\mathcal{PCO}$*s*, the total test at each node takes $O(m^3)$ time. Since, in the worst case, an $O(m^2)$ nodes are opened, the entire algorithm takes $O(m^5)$ time to construct the $\mathcal{WE}$-*Tree* and solve the *FeasblePath* problem.

**Q.E.D.** □

Let $n$ be the number of vertices in the typical *C-space* representation of the problem. The *WC* algorithm is expected to be faster than any *C-space* based method since, in the worst case, $n = O(m^3)$ (Latombe[7], Sharir[13]). This algorithm thus avoids the $O(m^3)$ time required to build the boundary description of *C-space*. Further, any *C-space* based algorithm is applicable to the $\mathcal{PCOR}$, since the *C-space* boundary can be computed from the $\mathcal{PCOR}$.

## 3.6 Results

The 3D *Window Corner* (*WC*) Algorithm, for planning translational paths, has been implemented in the C language. The program has been tested on several path planning examples on SUN SPARCstation 330 and IBM RISC 6000 machines. The input to the program consists of the description of $R$ and $\Phi$ in terms of the bounding faces, edges and vertices. The initial and final configurations of $R$ are also given. The program starts by computing the $\mathcal{PCOR}$ description of the problem. The initial position of $R$ corresponds to the origin, $O_{PCOR}$. This position also corresponds to the root node of the $\mathcal{WE}$-Tree. From the present position, all the polyhedral convex cones which form the $\mathcal{PCR}$ are generated, and all the *Window Edges* are identified. These cones are used to determine the reachability of each of the candidate *Window Corners* which are generated by the pair-wise intersection of the *Window Edges*. The reachable *Window Corners* are then appended to a list $OPEN$, which consists of all configurations of $R$ that have to explored further. This process iterates recursively, over each of the unexplored nodes in $OPEN$ until either the goal configuration is reachable from the present node or $OPEN$ does not contain any new nodes. This is equivalent to a breadth-first search and other techniques such as best-first or $A^*$ could be used to significantly improve the performance. In that case, a heuristic evaluation function could be defined which gives, in some sense, the proximity of a node to the goal configuration.

The *WC* algorithms is an example of the roadmap approach. The $\mathcal{WE}$-Tree has to be built only once for a given robot $R$, and obstacles, $\Phi$. Thereafter, for any start-goal

pair, the *FeasiblePath* problem can be solved simply, by connecting each of the terminal configurations to at least one of the *Window Edges* that are represented in the $\mathcal{WE}$-Tree.

We present the results of applying the 3D $\mathcal{WC}$ Algorithm to several assembly path planning problems. The graphic outputs are snapshots of dynamic demos which have been prepared. These demos are currently being displayed in a simple, portable MATLAB environment with the aid of additional programs that have been written to enable the viewing of 3D objects in motion, using a 2D MATLAB graphics display.

The performance of the program has been found to be satisfactory. However, problems of accuracy and robustness due to finite-precision arithmetic [20] could arise while the geometric (largely vector-based) operations are being carried out. A practical approach would be to "grow" the PCOR obstacles by an extremely small amount (say, an order of magnitude, or two, less than typical tolerance limits), before applying the algorithm. In this case, the algorithm would (theoretically) no longer be guaranteed to find a path if one exists, but any path that is reported will be guaranteed to be collision-free. This is equivalent to restricting the motion of $R$, so that it maintains a minimum safety distance from every obstacle.

### 3.6.1 A Cube in a Box Assembly

The initial configuration of this assembly consists of a cube which is placed in a box, Figure 10. The box has an opening on it's top face. The 3D $\mathcal{WC}$ Algorithm was used to plan a path which would move the cube to a goal configuration which has the cube outside and under the box, such that the top face of the cube is in planar contact with the lower face of the box. The feasible path found consisted of 5 motions. Of these, 3 were contact-motions and 2 were free-motions.

### 3.6.2 A Peg in a Blind-Hole Assembly

A cylindrical peg (shaft) is initially inserted into a blind-hole on the top face of a block. The blind-hole runs about two thirds of the height of the block. The 3D $\mathcal{WC}$ Algorithm was used to plan a path which removed the peg from the hole and placed it on the table (which is assumed to be supporting the block). Figure 11 shows the 2 contact-motions and 1 free-motion which were involved.

### 3.6.3 A Knuckle Joint Assembly

A polyhedral version of a knuckle joint assembly was designed. This assembly has five parts: the fork, eye, pin, collar and the taper-pin, all shown in Figure 12. The five parts are initially arranged on a table (assumed). The assembly process assumes that the fork is stationary while the rest are mobile. The 3D $\mathcal{WC}$ Algorithm was used to successively plan paths for the four assembly motions which are required to form the completed assembly. Figure 13 shows the assembly path computed for assembling the *eye* with the *fork*. Figure 14 shows the completed knuckle joint assembly. The remaining assembly motions which were computed, have not been shown due to space limitations.

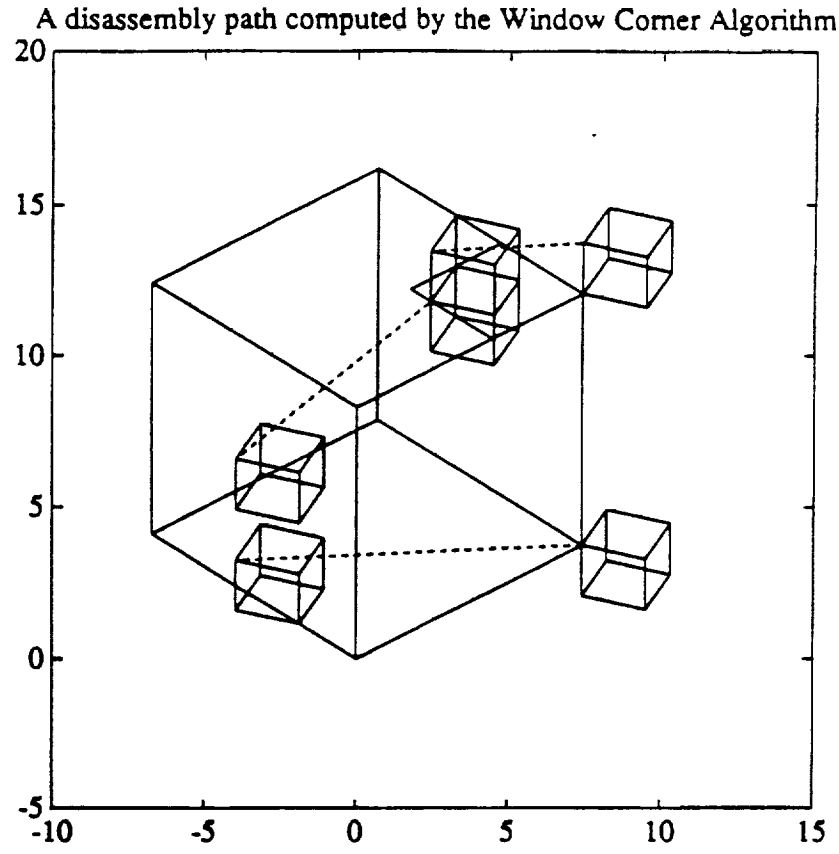


**Figure 10: Disassembly Path for a Cube in a Box Assembly, computed by the 3D *WC* Algorithm: 3 contact-motions, 2 free-motions**

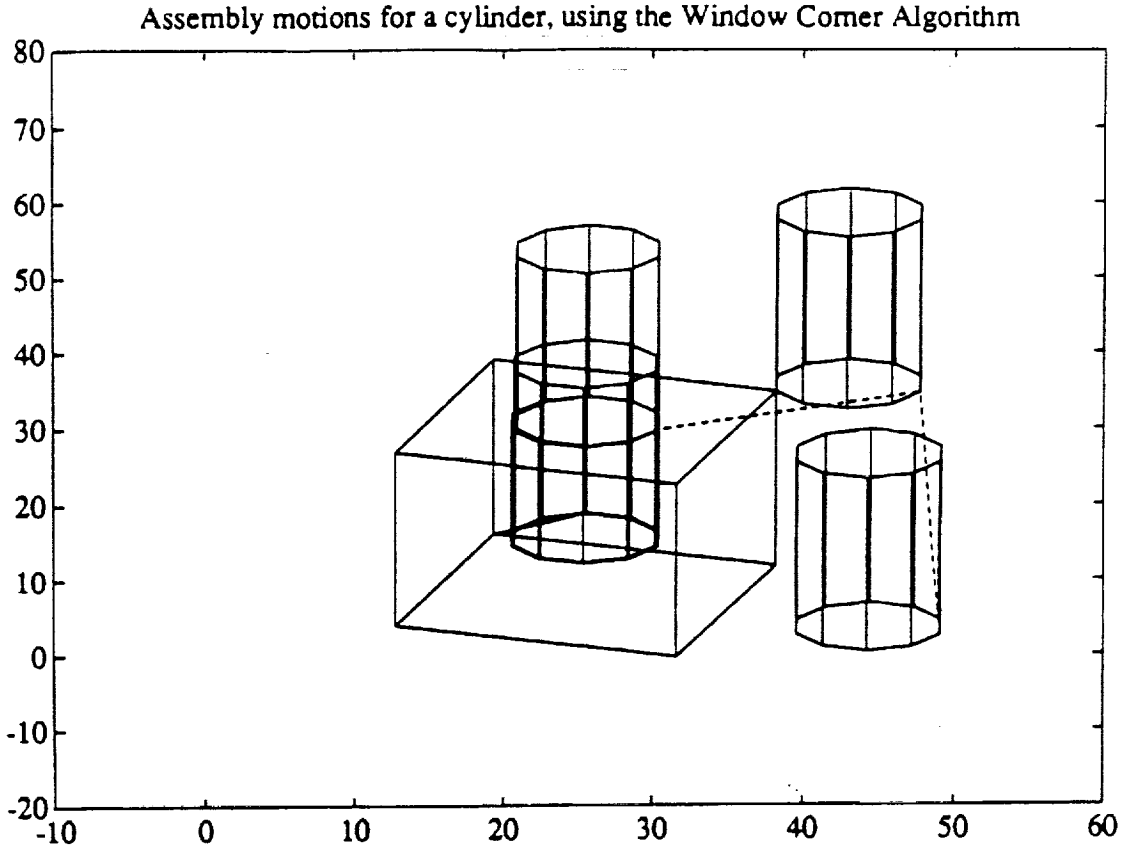


**Figure 11: Disassembly Path for a Peg in a Hole Assembly, computed by the 3D *WC* Algorithm: 2 contact-motions, 1 free-motion**

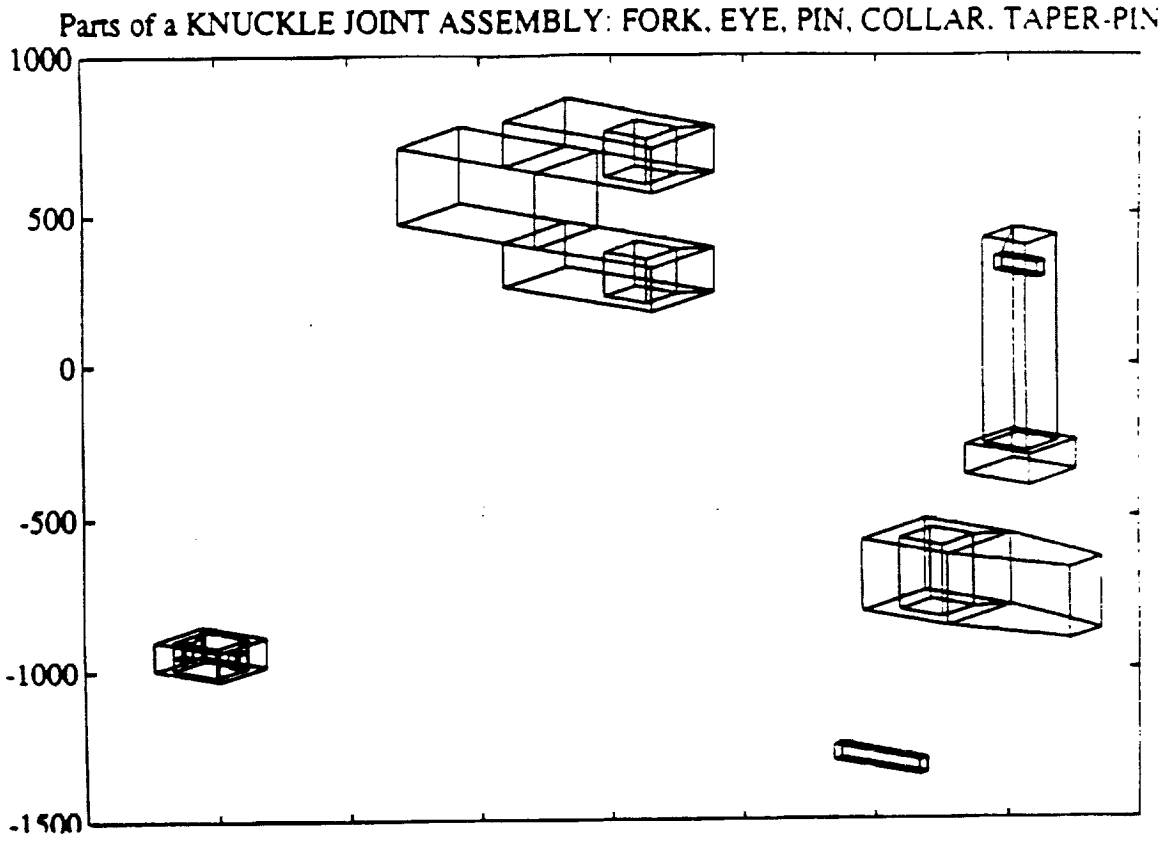


**Figure 12: Parts of a Knuckle Joint Assembly**

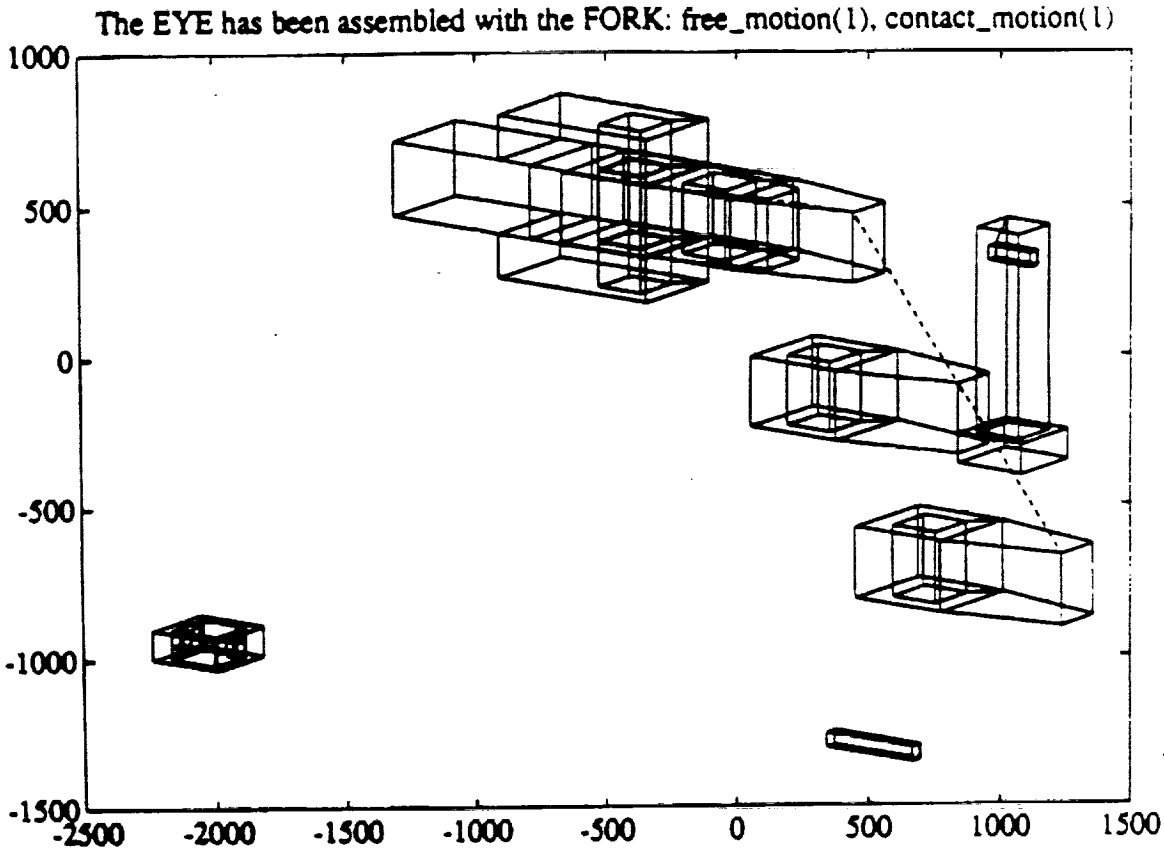


**Figure 13: Assembly Path Planning for a Knuckle Joint Assembly, using the 3D *WC* Algorithm**

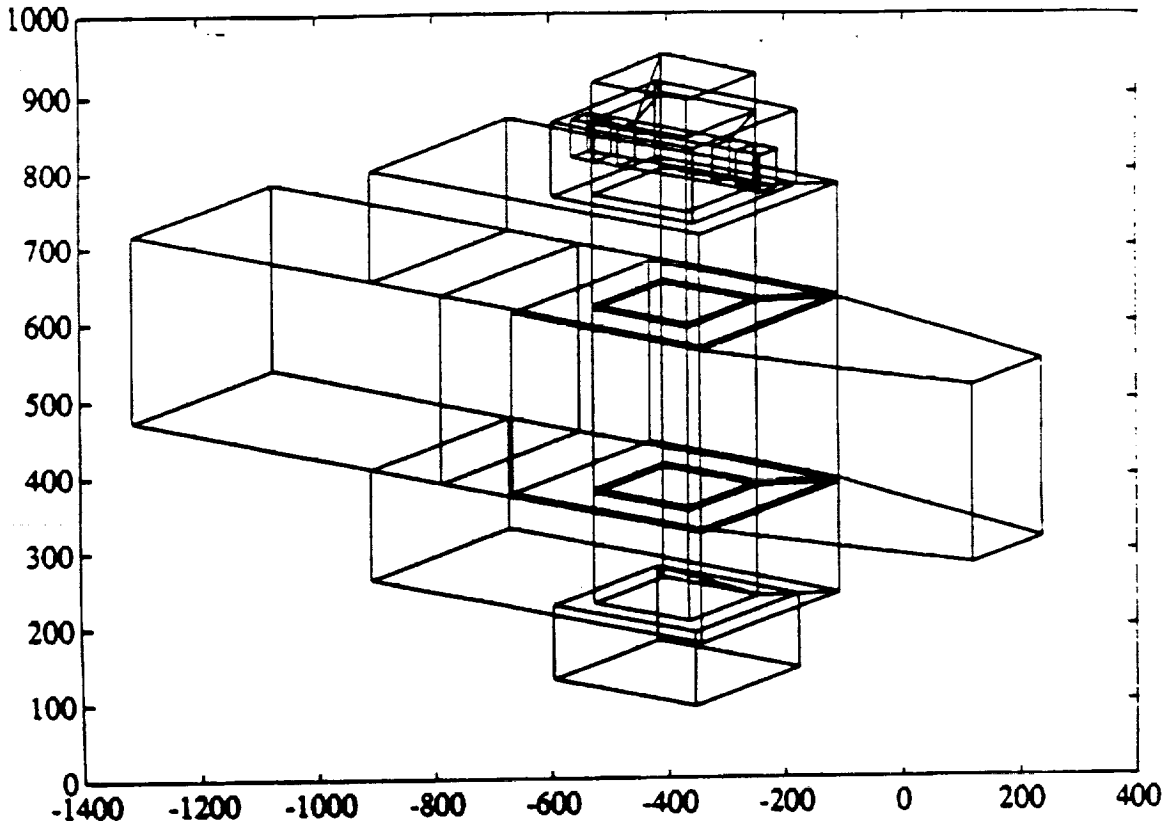


**Figure 14: The completed Knuckle Joint Assembly**

### 3.7 Minimum Distance between Non-convex Polyhedra

We show how the description of the $\mathcal{PCOR}$ can be used to compute the minimum distance between the robot and obstacle boundaries. This problem frequently arises in robot motion planning and collision avoidance problems. The methods proposed in the literature (Bobrow[15], Gilbert et al [16]) deal with convex polyhedra and utilize mathematical programming techniques. These algorithms were applicable to the non-convex case by considering the convex decompositions of 3D polyhedra, resulting in $O(m)$ time algorithms (without including the preprocessing time required for the 3D convex decomposition).

We can compute the minimum distance between $\mathcal{O}_{\mathcal{PCOR}}$ and a face of the $\mathcal{PCO}$s in constant time. Since there are an $O(m)$ $\mathcal{PCO}$s, the minimum distance between the robot and obstacle boundaries can be computed in $O(m)$ time. An intersection between the robot and obstacle boundaries is equivalent to $\mathcal{O}_{\mathcal{PCOR}}$ being located inside at least one of the $\mathcal{PCO}$s, and hence can be detected in $O(m)$ time.

Our algorithm is efficient for analytically computing the minimum distance between the boundaries of non-convex polyhedra, since it can be computed in $O(m)$ time and does not require the expensive convex decomposition of 3D polyhedra.

## 4 Conclusions

We have presented the correct and complete $\mathcal{WC}$ algorithm which searches for, translational paths among polyhedral subasemblies, even when the amount of free space tends to be severely limited. Polyhederal cones are used to represent geometrical translational constraints between the robot and obstacle boundaries and to support accurate and efficient reachability tests. We introduced *Window Corners* and *Window Edges* in the *Polyhedral Cone Representation* ($\mathcal{PCR}$) and this resulted in reducing the search space. Only a single connected component of the reachable space containing the initial placement of the moving object is searched. In two dimensions, a *Window Graph* with $k$ nodes and $O(k^2)$ edges, is built, and the shortest path found, in $O(km\log(m))$ time where $m$ is the product of the number of vertices describing the robot and obstacles respectively, $k = O(m)$ in the worst case and $1 \leq k \leq (m+2)$. In addition, we utilize the concept of a *Polyhedral Cone Obstacle Representation* ($\mathcal{PCOR}$) which transforms the problem, in $O(m)$ time, into that of a point moving amongst a collection, $O(m)$, of convex prism-shaped obstacles (in $E^2$ and $E^3$).

In three dimensions, The *3D Window Corner* 3D ($\mathcal{WC}$) algorithm constructs a *Window Edge* ($\mathcal{WE}$)*-Tree* in time $O(m^5)$. We search for feasible paths in a *hierarchical* manner. The algorithm determines the feasibility of incremental motion by using contact information only. Next, the feasibility of straight-line, and mutiple-step, assembly paths are evaluated. The low time complexity as a result of a finite search space, makes the *FeasiblePath* problem widely applicable, since many robotic applications would prefer fast computation of feasible paths, over exponential time for (continuous search space) shortest path algorithms. This trade-off is attractive for real-time operations such as telerobotic maintenance and repair, and for path planning during assembly sequence planning.

Since the B-rep. of the *C-space* obstcales is not constructed, both algorithms are efficient compared to path planners which first construct the *(C)-space*, an operation requiring time (and worst-case number of vertices) $O(m^2)$ and $O(m^3)$ in planar and polyhedral ver-

sions, respectively. Collectively, the above aspects result in algorithms which search for feasible paths based on visiting *Window Corners* only, and which can be implemented with ease in a wide range of robotic and assembly sequence planning situations. In addition, we showed how the $\mathcal{PCOR}$ can be used to efficiently compute the minimum distance between the boundaries of a pair of non-convex 3D polyhedra, without requiring a 3D convex decomposition. Results obtained for planar and polyhedral assembly path problems were presented.

### Acknowledgements

This work is supported by the New York State Center for Advanced Technology (CAT) in Automation and Robotics at Rensselaer Polytechnic Institute and the NASA Center for Intelligent Robotic Systems for Space Exploration. The CAT is funded by a grant from the New York State Science and Technology Foundation.

## References

[1] T. Asano, T. Asano, L. Guibas, J. Hershberger, H. Imai, "Visibility of Disjoint Polygons". In *Algorithmica*, 1(1), pp. 49-63, 1986.

[2] S.K. Ghosh and D.M. Mount, "An Output Sensitive Algorithm for Computing Visibility Graphs". In *Proc. of the 28th IEEE Symposium on Foundations of Computer Science*, pp. 11-19, 1987.

[3] L.S. Homem de Mello and A.C. Sanderson, "Representations of Mechanical Assembly Sequences". In *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 2, pp. 211-227, 1991.

[4] L.S. Homem de Mello and A.C. Sanderson, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences". In *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 2, pp. 228-240, 1991.

[5] S.S. Krishnan and A.C. Sanderson, "Reasoning About Geometric Constraints for Assembly Sequence Planning". In *Proc. of the IEEE International Conference on Robotics and Automation*, Sacramento, CA, pp. 776-782, 1991.

[6] S.S. Krishnan and A.C. Sanderson, "The Window Corner Algorithm for Robot Path Planning with Translations". In *Proc. of the IEEE International Conference on Robotics and Automation*, Nice, France, in press, 1992.

[7] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[8] Y. Liu and S. Arimoto, "Proposal of Tangent Graph and Extended Tangent Graph for Path Planning of Mobile Robots". In *Proc. of the IEEE International Conference on Robotics and Automation*, Sacramento, CA, pp. 312-317, 1991.

[9] T. Lozano-Pérez and M.A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles". In *CACM*, Vol. 22, No. 10, pp. 560-570, 1979.

[10] T. Lozano-Pérez, "Spatial Planning: A Configuration Space Approach". In *IEEE Transactions on Computers*, Vol.C-32, No.2, pp. 108-119, 1983.

[11] C. Papadimitriou, "An Algorithm for Shortest-path Motion in Three Dimensions". In *Information Processing Letters*, Vol. 20, No. 5, pp. 259-263, 1985.

[12] M. Sharir and A. Schorr, "On Shortest Paths in Polyhedral Spaces". In *SIAM Journal of Computing*, Vol. 15, No. 1, pp. 193-215, 1986.

[13] M. Sharir, "Efficient Algorithms for Planning Purely Translational Collision-free Motion in Two and Three Dimensions". In *Proc. of the IEEE International Conference on Robotics and Automation*, Raleigh, NC, pp.1326-1331, 1987.

[14] E. Welzl, "Constructing the Visibility Graph for n-Line Segments in $O(n^2)$ time". In *Information Processing Letters*, 20, pp.167-171, 1985.

[15] J.E. Bobrow, "A Direct Minimization Approach for Obtaining the Distance between Convex Polyhedra". In *The International Journal of Robotics Research*, Vol. 8, No. 3, pp. 65-76, 1989.

[16] E.G. Gilbert, D.W. Johnson and S.S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space". In *IEEE Journal on Robotics and Automation*, Vol. 4, No. 2, pp. 193-203, 1988.

[17] Y.K. Hwang and N. Ahuja, "A Potential Field Approach to Path Planning". In *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 1, pp. 23-32, 1992.

[18] R.H. Wilson and J.-F. Rit, "Maintaining Geometric Dependencies in an Assembly Planner", In *Proc. of the IEEE International Conference on Robotics and Automation*, Cincinnati, OH, pp. 890-895, 1990.

[19] D.F. Baldwin, T.E. Abell, M-C.M. Lui, T.L. de Fazio and D.E. Whitney, "An Integrated Computer Aid for Generating and Evaluating Assembly Sequences for Mechanical Products", In *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 1, pp. 78-94, 1991.